■ PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

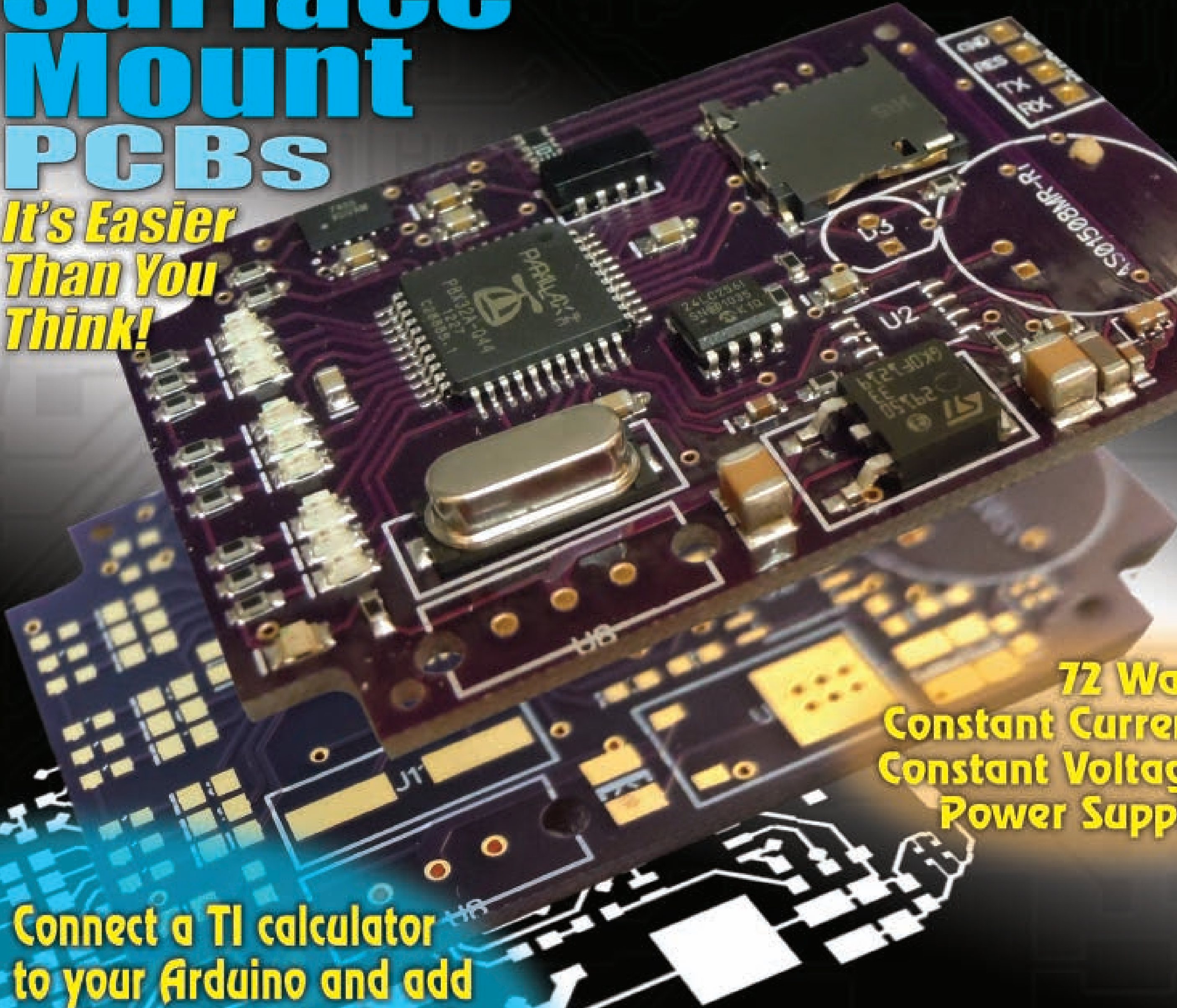
NUTS AND VOLTS

EVERYTHING FOR ELECTRONICS

www.nutsvolts.com
May 2014

Make Your Own Surface Mount PCBs

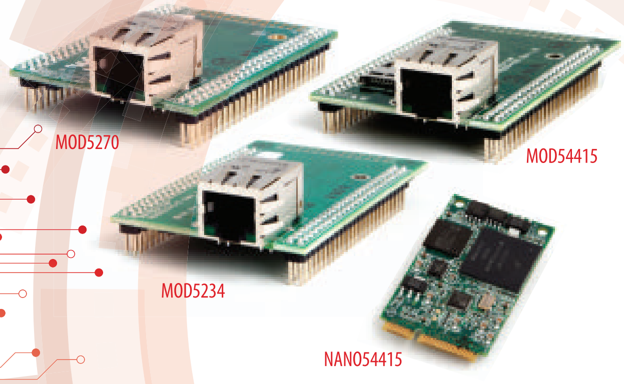
It's Easier Than You Think!



72 Watt Constant Current Constant Voltage Power Supply

Connect a TI calculator to your Arduino and add math functions and easier graphic capabilities.

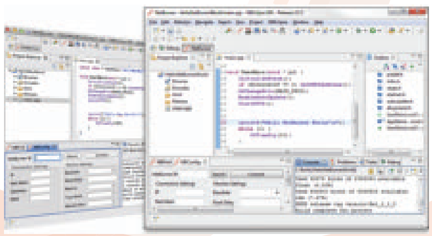
Ethernet Core Modules with
High-Performance Connectivity Options



- **MOD5270**
  147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO
  3 UARTs · I²C · SPI
- **MOD5234**
  147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs
  I²C · SPI · CAN · eTPU (for I/O handling, serial communications, motor/timing/engine control applications)
- **MOD54415**
  250 MHz processor with 32MB flash & 64MB RAM · 42 GPIO · 8 UARTs
  5 I²C · 3 SPI · 2 CAN · SSI · 8 ADC · 2 DAC · 8 PWM · 1-Wire® interface
- **NANO54415**
  250 MHz processor with 8MB flash & 64MB RAM · 30 GPIO · 8 UARTs
  4 I²C · 3 SPI · 2 CAN · SSI · 6 ADC · 2 DAC · 8 PWM · 1-Wire® interface

Add Ethernet connectivity to an existing product, or use it as your product's core processor

The goal: Control, configure, or monitor a device using Ethernet



The method: Create and deploy applications from your Mac or Windows PC. Get hands-on familiarity with the NetBurner platform by studying, building, and modifying source code examples.

The result: Access device from the Internet or a local area network (LAN)

The NetBurner Ethernet Core Module is a device containing everything needed for design engineers to add network control and to monitor a company's communications assets. For a very low price point, this module solves the problem of network-enabling devices with 10/100 Ethernet, including those requiring digital, analog and serial control.

MOD5270-100IR........$69 (qty. 100)	NNDK-MOD5270LC-KIT..............$99
MOD5234-100IR........$99 (qty. 100)	NNDK-MOD5234LC-KIT............$249
MOD54415-100IR.....$89 (qty. 100)	NNDK-MOD54415LC-KIT..........$129
NANO54415-200IR...$69 (qty. 100)	NNDK-NANO54415-KIT...............$99

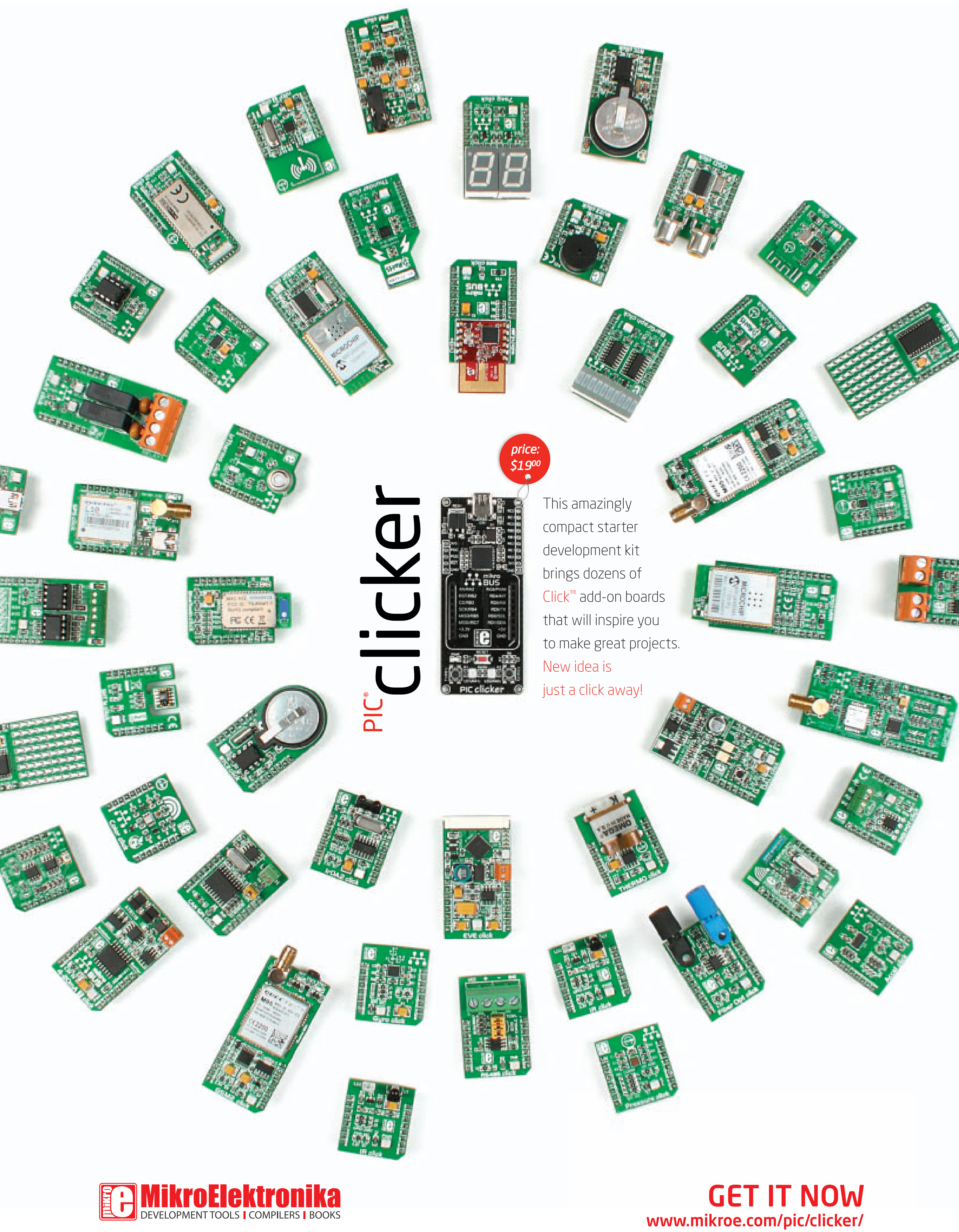
NetBurner Development Kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kits include all the hardware and software you need to build your embedded application.

- **For additional information please visit**
  http://www.netburner.com/kits

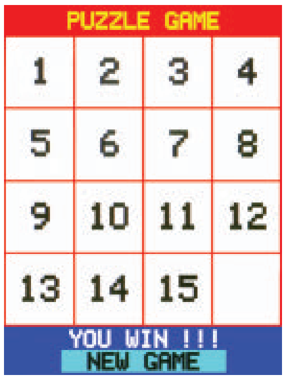
NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member


price: $19.00
PIC® clicker
This amazingly compact starter development kit brings dozens of Click™ add-on boards that will inspire you to make great projects.
New idea is just a click away!
PIC clicker
MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS
GET IT NOW
www.mikroe.com/pic/clicker/

May 2014



Columns

Departments

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

DEVELOPING PERSPECTIVES

When Additive and Subtractive Technologies Don't Add Up

When I made the move to 3D printing a few years ago, I imagined that by now I would have sold my milling machine, drill press, and other 'subtractive' technologies. The reality is that 3D printing is simply another tool in my prototyping toolbox. This additive technology isn't really clean or waste free — I'm constantly vacuuming starter strands of PLA and ABS from my shop floor. Plus, the technology isn't completely safe. There's the hot extruder, hot bed, and — as I recently discovered — danger in machining a 3D printout.

I like to prototype in PLA and then use ABS for a final print. Printing with PLA has a number of advantages over printing with ABS: It's relatively odorless; the nozzle temperature is relatively low; there's no need to heat the printing platform (a 10 minute process); and there's no fumbling with the plastic film tape that always traps a few bubbles. I just put down a layer of blue painter's tape on the platform and hit the print button. Within a minute, I'm printing.

I typically take the PLA printout and rework it with a Dremel or other tool to reshape or add mounting holes to the prototype. That is, until now. For some time, I've noticed prickly feelings in my fingertips, akin to what I've experienced after working with fiberglass insulation. I didn't connect the 3D printing with the pain until I switched from natural to black PLA.

Under my workstation microscope, I found dozens of short black PLA shards embedded in my fingertips. After spending an hour with tweezers extracting the splinters, I decided that — as far as PLA is concerned — additive and subtractive technologies don't add up.

I haven't given up on PLA altogether because of the advantages noted above, but when there's a modification called for I do it in software and make a new print. I still rely on my drill press, Dremel, and other subtractive tools to get a prototype into shape, but now only with an ABS plastic printout.

My latest experiment in 3D printing is thermoplastic elastomer (TPE) — basically printable rubber. It's expensive at about double the price of PLA or ABS ($50/500 gm from AdaFruit, not including shipping). However, the ability to print flexible structures is intriguing, and — like ABS — there aren't any fragile shards flying about when I shave an edge or drill a few holes in the printout.

Clearly, 3D printer technology is evolving, and the spools of PLA, ABS, TPE, and other materials will eventually give way to the powders and liquids used in commercial printers.

So, should you wait for those next-generation materials to reach the consumer 3D market? No way. 3D printing is a fantastic prototyping tool that should be part of every electronics experimenter's arsenal. Just take the proper precautions when you work with printouts, such as gloves and eye protection when you're reworking something printed in PLA. **NV**

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

VP of Operations
Vern Graner
vern@nutsvolts.com

CONTRIBUTING EDITORS
Joe Pardue, John Gavlik, Martin Hebel, Dave Dorhout, Garry Spencer, William Massano, Allen Ripingill, Jon McPhalen, Jeff Eckert, Paul Verhage

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@nutsvolts.com

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION
Sean Lemieux

READER FEEDBACK

Random Thoughts

I read the Techknowledgey 2014 column in the March issue with great interest. I have been wearing such a device on my wrist for the past 12 years; it is the Seiko Kinetic watch.

If you listen carefully, you can hear the mechanism spinning a generator (with 'random' wrist motion) which charges a battery, which powers an analog watch (which incidentally keeps amazingly good time: ± one second in 90 days!). It is made of titanium and is not bulky.

The WITT generator is a very nice device and will certainly have many applications — particularly at sea.

Richard Turner KA8HXR

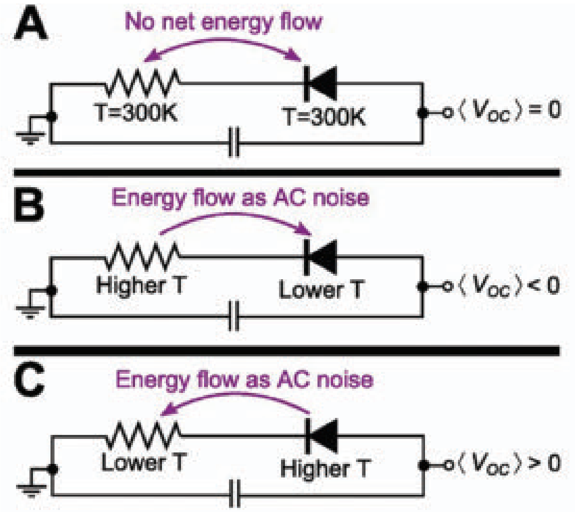
Yeah, I'd forgotten about self-winding watches. I seem to remember having one when I was a kid. I think it broke after a couple weeks.

I have always taken issue with physics teacher's insistence that perpetual motion is physically impossible. I suppose eternal perpetual motion is, as you have to consider friction, the second law of thermodynamics, and so forth. You can certainly play forces like gravity and inertia against each other for a long, long time, however. I'm pretty sure that's how the solar system works.

Many years ago, a sailor buddy explained to me that a sailboat can actually travel faster than the wind

Continued on page 52

ADVANCED TECHNOLOGY



■ Diode-resistor generator circuits.

Photo courtesy of Federico Capasso and PNAS.

Reverse Solar Generation?

One of the inherent problems with solar power is that the sun has a tendency to set every night, so it can be exploited for a limited and highly variable number of hours on any given day in any given location. You can store generated energy in batteries, of course, to provide a constant output, but what if you could reverse the process and tap into the earth's infrared emissions? Physicists at the Harvard School of Engineering and Applied Sciences (**www.seas.harvard.edu**) in a paper recently appearing in the Proceedings of the National Academy of Sciences, have explained how it might be done someday.

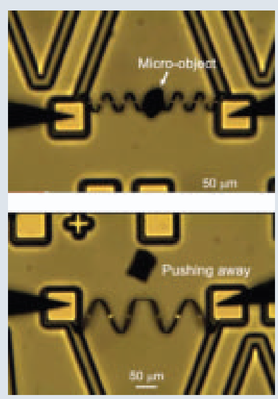
One method is based on a concept explained back in 1968 by J. B. Gunn, inventor of the Gunn diode. As shown in the above diagram, the flow of AC noise in a diode-resistor generator circuit depends on the temperature input. At thermal equilibrium (A), no current is generated. In circuit B — with a higher temperature in the resistor — current flows as in conventional rectifier operation. In circuit C, however, with the diode at a higher temperature, energy flows in the opposite direction. According to the research team, "the role of the resistor could be played by a microscopic antenna that very efficiently emits the earth's infrared radiation toward the sky, cooling the electrons in only that part of the circuit." As a result, "you get an electric current directly from the radiation process, without the intermediate step of cooling a macroscopic object." According to the paper, a single device could be coated in many such circuits, pointed at the sky, and used to generate power.

Alas, serious hurdles exist. According to lead author Steven J. Byrnes, "The more power that's flowing through a single circuit, the easier it is to get the components to do what you want. If you're harvesting energy from infrared emissions, the voltage will be relatively low. That means it's very difficult to create an infrared diode that will work well." In addition, "Only a select class of diodes can switch on and off 30 trillion times a second, which is what we need for infrared signals. We need to deal with the speed requirements at the same time we deal with the voltage and impedance requirements." In other words, don't hold your breath. ▲

Microscale Muscle

One of the materials currently being investigated for advanced electronic operations is vanadium dioxide which is known for its rare quality of being an insulator at low temperatures, but a metal-like conductor at levels above about 153°F (67°C). It is also known for its strange ability to vary its size, shape, and physical characteristics, and a group of researchers at the Lawrence Berkeley National Lab (**www.lbl.gov**) have exploited that characteristic to create a "micro-sized robotic torsional muscle/motor." Reportedly, for its size, it is 1,000 times stronger than a human muscle. In only 60 ms, it can push objects 50 times its own weight over a distance five times its length. According to physicist Junqiao Wu, "We've created a micro-bimorph dual coil that functions as a powerful torsional muscle, driven thermally or electro-thermally by the phase transition of vanadium dioxide. Using a simple design and inorganic materials, we achieve superior performance in power density and speed over the motors and actuators now used in integrated micro-systems."

The micro-muscle is in the form of a ribbon of chromium and vanadium dioxide that forms a helical dual coil with each end connected to chromium electrode pads. When warmed (electrically or optothermally), the material contracts along one dimension while expanding along the other two, thus creating the muscular effect. It can also act as a proximity sensor in which a rapid change in the muscle's resistance and shape is used to push the detected object away. According to a published paper, the micro-muscles have operated for over one million cycles with no degradation and have shown rotational speeds up to 200,000 RPM, amplitude of 500 to 2,000 degrees per millimeter of length, and energy power densities up to 39 kW/kg. No specific applications were cited, but Wu noted, "With its combination of power and multi-functionality, our micro-muscle shows great potential for applications that require a high level of functionality integration in a small space." ▲



■ Vanadium dioxide micro-muscle pushes away an object.

COMPUTERS and NETWORKING

Feel More Secure

■ The Blackphone runs a security-oriented version of Android.

Your phone may be amazingly smart, but it's probably pretty dumb when it comes to security issues. Its memory contains extensive personal data including email addresses, photos, financial information, and other private details, so there's a lot in there that appeals to hackers, phishers, and other vermin. Even though a range of security apps are available, various sources have reported that up to 80 percent of smartphone users have installed no malware protection at all. If you are among them, it is highly advisable that you take steps to foil would-be attackers. The FCC (Federal Communications Commission) actually offers a free smartphone security checker at **www.fcc.gov/smartphone-security** which allows you to select your operating system and follow 10 customized steps to protect your mobile device.

If you want to be really safe (or are too lazy to implement security measures), maybe you should pick your next phone from among a few models that have been designed from the ground up to ensure your privacy and security. One of the latest and securest is the Boeing Black, but unless you are among "government agencies and companies engaged in contractual activities with those agencies that are related to defense and homeland security," you are persona non grata. S.O.L., we might say. Never fear, though, because a similar device is the result of a joint venture involving Silent Circle (**www.silentcircle.com**) and Geeksphone (**www.geeksphone.com**). The company's Blackphone is billed as "the world's first smartphone placing privacy and control directly in the hands of its users." The phone — which runs on a security-oriented Android OS called PrivatOS — is a carrier- and vendor-independent unit that allows users to securely make and receive calls, exchange texts, transfer and store files, and engage in video chats. Reportedly, it is compatible with any carrier except Verizon. Note that it is not an NSA-proof device, so if you're a terrorist, they can still get you. Details about the $629 unit are available at **www.blackphone.ch**. ▲

Dig Up the Dirt!

So, your next-door neighbor seems pleasant. Dresses well, drives an expensive car, immaculately maintains the yard and pool, and smiles and waves when you drive by. That's nice, but how do you *really* know that it isn't all a façade, concealing the fact that you're living next to an insane dangerous criminal like John Wayne Gacy, Charlie Manson, or Justin Bieber? No problem! Look him up on Dirt Search (**www.dirtsearch.org**). Just enter the first name, last name, and state, and Dirt Search does the rest. Unlike other search services, it doesn't demand payment for the info. It's absolutely free (although donations are happily accepted). The odd thing is that it takes about five minutes to complete a search for one person. This makes it seem like it must be very thorough, but I did a similar Google search that turned up 118 million results in just over half a second, so what's up? Who knows? Maybe Dirt Search is based on an IBM PC XT with a 300 baud modem. In any event, you might find something interesting. ▲

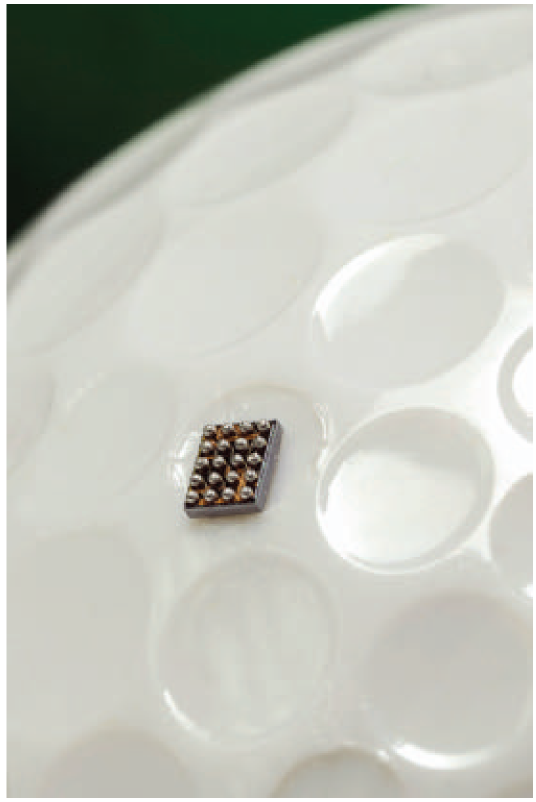
■ Could a malevolent miscreant be your neighbor?

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/may2014_TechKnow14**.

CIRCUITS and DEVICES

Honey, I Shrunk the MCU

Billed as the world's smallest and most energy-efficient ARM-based MCU, the new Kinetis KL03 from Freescale Semiconductor (**www.freescale.com**) builds on the older KL02, adding more features, advanced integration, and easier implementation (a ROM-based boot loader allows factory programming and only firmware upgrades) that is stuffed into a 1.2 x 2.0 mm package. This makes it 15 percent smaller than the KL02 and 35 percent smaller than other competing 32-bit chips. The product is aimed at space-sensitive applications in consumer, healthcare, and industrial markets, including Internet of Things devices. Among other things, the device integrates a 48 MHz Cortex-M0+ core (1.17V to 3.6V operation), 32 KB of Flash memory, and 2 KB of RAM, 8K ROM, a 12-bit ADC, and an internal voltage reference for high ADC accuracy. It also features a secure real time clock and timers for applications including motor control. Samples are available now, with full production scheduled for June. Buy 100,000 of them, and they'll run $0.75 each. Prices are likely to be considerably higher in smaller quantities. ▲

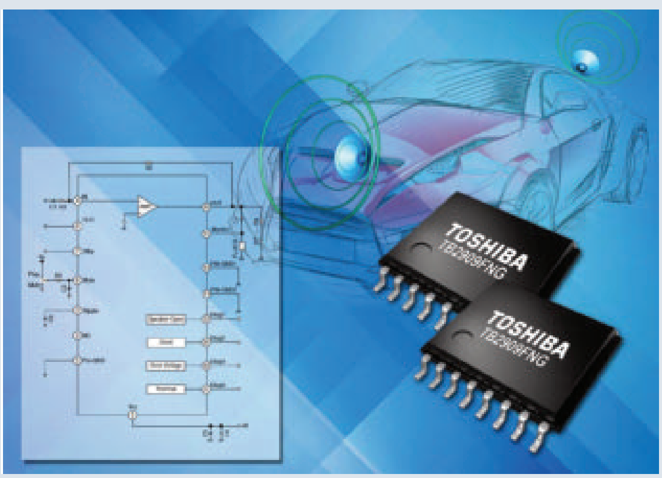


■ Freescale's KL03 MCU — smaller than a dimple in a golf ball.

More Noise, Please

If one of the things you like about your electric or hybrid car is the pleasant lack of engine noise, don't get too used to it. Toshiba (**www.toshiba-components.com**) recently announced the TB2909FNG — a single-chip amplifier designed to add the sound of an internal combustion engine to EVs and HEVs. Yes, you read that right. You see, in 2011 the European Commission adopted a measure stating that manufacturers shall be required to install an "acoustic vehicle alerting system" (AFAS) in these hybrids, that the sound should be variable to reflect vehicle behavior, and that they should sound similar to a vehicle of the same category equipped with an internal combustion engine. It is, of course, intended to enhance safety and prevent drivers from running over cyclists, pedestrians, billy goats, and whatever else strays into the roadway.

Even though the requirement has no legal status in the US, it is not clear that imports won't come equipped that way, that it will be simple to disable the devices, or that the National Highway Traffic Safety Administration won't decide it's a great idea. In any event, the TB2909FNG will run on 6V to 16V, delivering a maximum output of 5W. (Is this enough to simulate engine noise for a Tesla springing from 0 to 60 in 4.2 seconds?) Total harmonic distortion is rated at 0.08 percent, so you should be able to detect those ersatz valves pinging. ▲



■ Toshiba's new power amplifier, designed to create simulated engine noise.

INDUSTRY and the PROFESSION

Still Below the Surface (of Profitability)

Microsoft jumped into the tablet market in June 2012 with the original Surface, which was followed up last October with the Surface 2 and Surface Pro. Sales have been so underwhelming that the company was actually sued by several law firms who accused it of misleading stockholders about the sales levels, which they deemed an "unmitigated disaster."

According to *Computerworld*, Microsoft posted increased sales revenues for the last quarter of 2013 ($893 million) but — as shown on its SEC Form 10-Q — lost $39 million on the devices. It appears to be a case of "too little, too late" to compete in an already crowded market. On the bright side, Bill Gates is the richest man in the world again, with $76 billion in his piggy bank. **NV**

■ Microsoft's Surface tablet is still a financial loss.

The Arduino Classroom
Arduino 101 — Chapter 5: Analog Output

Last month, we continued with the Arduino 101 Classroom series and discussed digital input using a momentary pushbutton. We also learned some more software concepts, and then ended with a lab that combined what we learned about LEDs and pushbuttons to create a reaction time tester to analyze how quickly you can press a button after an LED turns off. This month, we will first learn how to use the Arduino serial library and the serial monitor to communicate between the Arduino and a PC — something we'll need for setting values for analog output. After that, we will discuss the differences between analog and digital, then learn to output analog signals to control the brightness of an LED and the angular position of a servomotor. All of this will add some very useful tools to your personal computing and electronics toolkit.

Talking to the PC

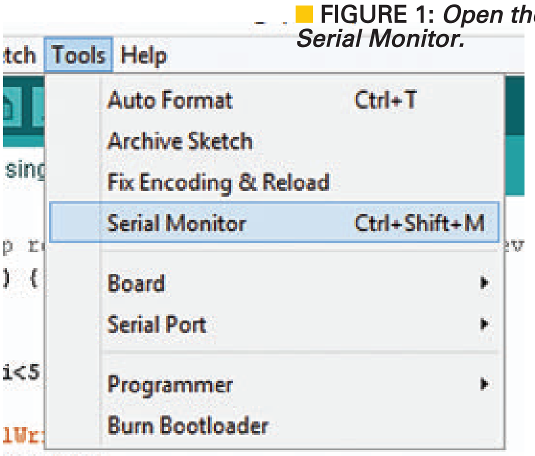
In this chapter, we are going to learn mainly about analog output but first let's take a look at a very handy Arduino class of functions: the serial class. We used a couple of functions from that class at the end of Chapter 4: *the serial.print()* and *serial.println()* functions. This month, we'll discuss these in a bit more detail and add a few more useful serial class functions to our arsenal.

Being able to communicate with an Arduino via a PC terminal program is very handy indeed. It allows us to transmit information from the Arduino to the PC by using the Arduino IDE's built-in serial monitor. It allows us to receive commands and data from the PC to an Arduino program. Doing this lets us open a window to see into the Arduino that can be vital for testing software concepts and for debugging programs — both you will get very acquainted with as we go along.

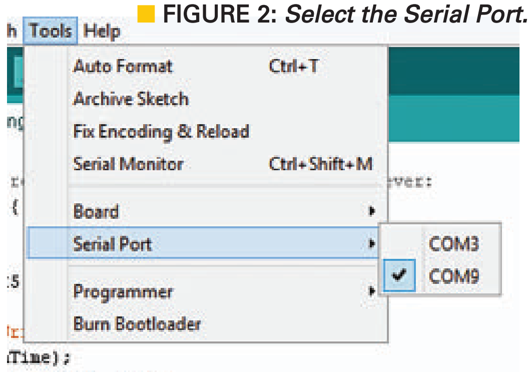
Sending Text to the PC

When we program the Arduino, we use the IDE (Integrated Development Environment) discussed in Chapter 1. The IDE uses the USB connection to do the communication through a serial port with a COM# identifier for the PC input port that — as we saw— lets us select from the Tools menu. The IDE takes care of all the communication between the IDE and the Arduino to allow us to upload programs that we write on the PC to run on the Arduino.

We can also use this communications channel to send and receive data to/from the PC by using the



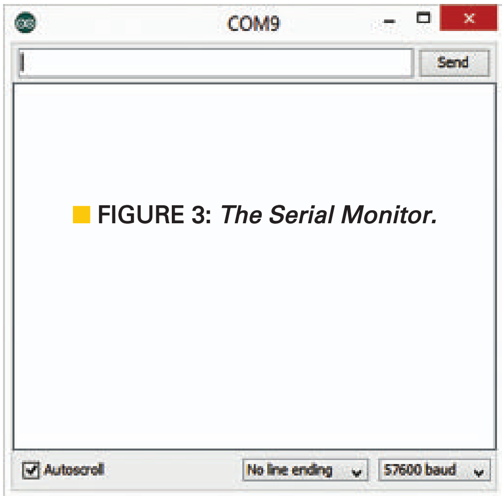
■ FIGURE 1: *Open the Serial Monitor.*



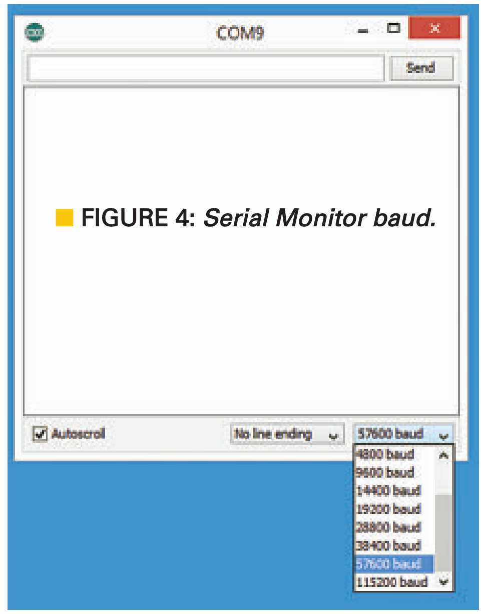
■ FIGURE 2: *Select the Serial Port.*

serial monitor.

We open the serial monitor by going to the Tools menu in the Arduino IDE and clicking on it as shown in **Figure 1**. Make sure you have selected the correct COM port as in **Figure 2** (this was discussed in Chapter 1). You'll see a new serial monitor window open as shown in **Figure 3**.



■ FIGURE 3: *The Serial Monitor.*

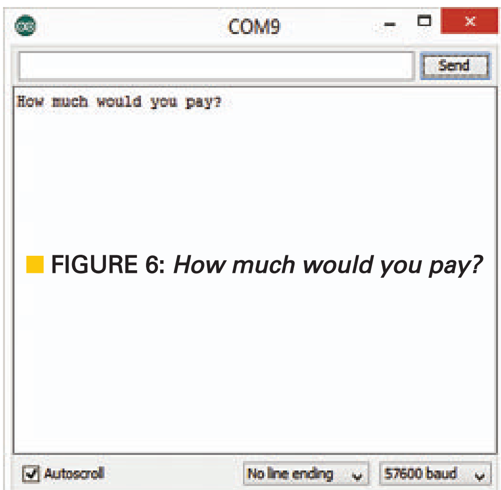


■ FIGURE 4: *Serial Monitor baud.*

Setting Up Our Serial Communication/Baud Rate

We must set up the serial port in the *setup()* function in our Arduino program. To do this, we use the *Serial.begin(baud)* function where the 'baud' (sometimes referred to as baud rate) is a number that is proportional to the speed of the communication. The bootloader on the Arduino Uno is set to 57600 baud, so it makes sense to set all our communications to that rate. Use the following in the *setup()* function:

```
Serial.begin(57600);
```



■ FIGURE 6: *How much would you pay?*

You will want to verify that you've set the serial monitor to the same baud. Do this by opening the baud drop-down selector in the serial monitor as shown in **Figure 4**.

■ FIGURE 5: *Hello World.*

Using the Serial Class

Serial is an Arduino class that contains many functions such as *print()* and *println()*. The difference between these two functions is that *Serial.println()* adds a line feed to the end of the text it sends to the serial monitor; that line feed causes the serial monitor to print the next text on the next empty line of the display.

Each of these functions takes a string of text delimited by quotes such as "This is a string of text" and sends that text to the serial monitor. [These functions can also take a second parameter that can be used to translate numbers into different bases, but this is beyond the scope of our discussion at this point.] If we write:

```
void setup()  {
  Serial.begin(57600);
}
void loop()  {

  Serial.println("Hello, world!");
  Serial.println();
  Serial.print("How ya doin?");

  for(;;);
}
```

when we upload and run the sketch, we will see the text on the serial monitor as shown in **Figure 5**.

Receiving Text from the PC

Let's learn to receive numbers from the PC. These numbers will be of the *int* (integer) datatype. On the Arduino, they can be any value from 0 to 65535. We will use the *parseInt()* function from the serial class as follows:

```
void setup()  {
  Serial.begin(57600);
```

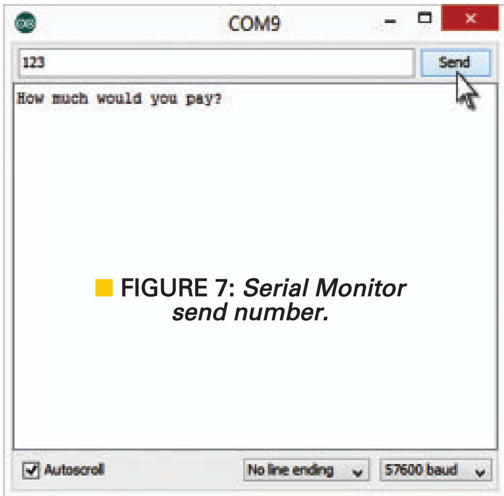


FIGURE 7: *Serial Monitor send number.*

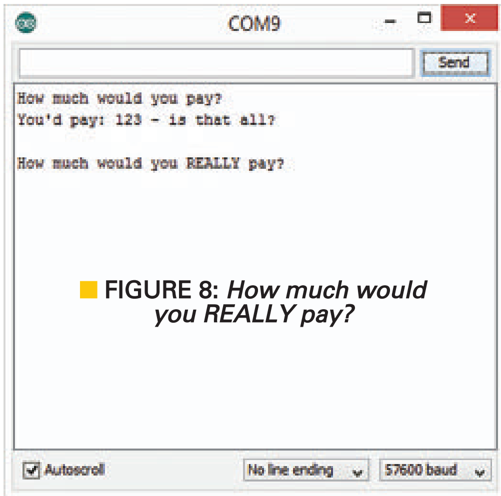


FIGURE 8: *How much would you REALLY pay?*

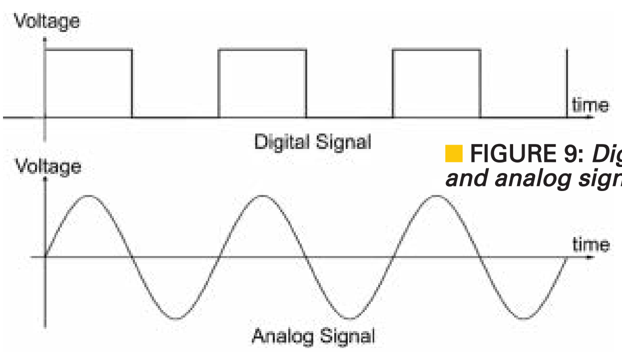


FIGURE 9: *Digital and analog signals.*

```
  Serial.println("How much would you pay?");
}

void loop() {
  int pay;

  if(Serial.available()) {
    pay =  Serial.parseInt();

   Serial.print("You'd pay: ");
   Serial.print(pay);
   Serial.println(" - is that all?");
   Serial.println();
   Serial.println("How much would you REALLY
   pay?");
   Serial.println();
  }
}
```

When the program first runs, we see the text "How much would you pay?" that was sent by the *Serial.println* in the *setup()* function as shown in **Figure 6**.

First, we enter an integer into the textbox at the top of the serial monitor such as '123,' then click Send as shown in **Figure 7**.

The *loop()* has an *if* statement that uses the *Serial.available* function to check to see if the PC is sending anything. If something has been sent, the *Serial.parseInt* function converts the data from the PC into an integer variable *pay*. Then, several *Serial.print* and *Serial.println* functions are used to display the number that was sent and to challenge the users as to what they would really pay as shown in **Figure 8**.

The message will repeat each time you input a new amount. We will use these concepts later in the chapter in the labs that let us send numbers to set LED brightness and servomotor positions.

As a little note of warning, if you open the serial monitor and instead of seeing the expected readable text you see a bunch of nonsense characters, check your baud. Both your Arduino program and the serial monitor must be using the same baud rate (57600, in our example) or the transmission and reception will be garbled. This is a very common error, so be prepared. Now, let's learn about analog output.

Analog Versus Digital

A **signal** is a quantity we detect that changes over time. We will look at signals for voltages that do just that.

The term **digital signal** refers to two discrete voltage states changing over time. These states represent a higher and lower voltage. The higher voltage in our system is +5 volts, while the lower voltage is zero volts (also known as ground; sometimes shown as GND). Digital signals are what computers use to operate.

The term **analog signal** refers to continuously varying voltage over time. For our system, these signals will vary between +5 volts and zero volts. Digital signals have only two states — on or off — while analog states are continuous so, in essence, have infinite possible values; in our case, between +5 and zero. **Figure 9** shows examples of digital and analog signals.

Analog Output Using PWM (Pulse Width Modulation)

We've learned that the Arduino can output digital signals on some of its pins. What we will learn this time is that several Arduino pins can output digital signals that can be used to create analog signals. These pins are marked with a '~' on the Arduino Uno board, and are numbered: 3, 5, 6, 9, 10, and 11 as shown in **Figure 10**.

The Arduino does not actually produce a variable continuous voltage on its analog pins; it produces **pulses** of digital voltages that can be averaged out by external

devices so that they produce results functionally similar to a variable voltage.

Figure 11 illustrates this concept with four **pulse trains** (pulses that repeat over a set interval know as the **period**) where the vertical represents voltage (zero to +5) and the horizontal represents time. In the first pulse train, the pulse is turned on to +5 volts 10% of the time and off to zero volts 90% of the time. In that case, the average voltage over time is 10% of five volts or 0.5 volts.

The second pulse train is 30% on and 70% off, giving an average of 1.5 volts. The third pulse is 50% on and 50% off, and averages to 50% of five volts or 2.5 volts. The last one is 90% on and 10% off for a 4.5 volts average. [Please note this is a simplification of what happens in real electronic systems and is meant to help understand what is going on at this point in the discussion.]

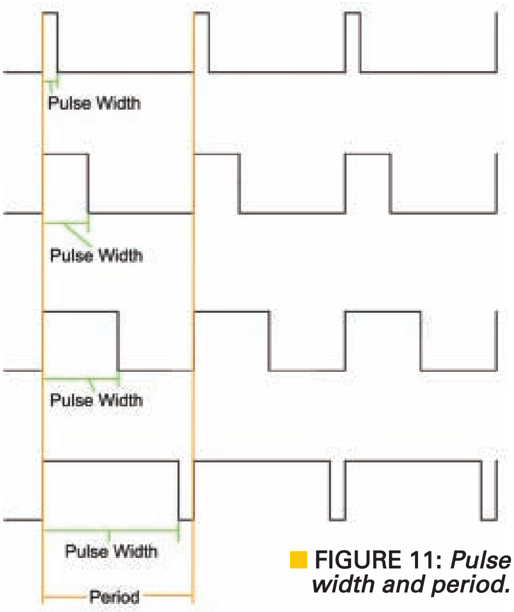
These digital signals are known as **PWM** (**Pulse Width Modulation**). To output PWM signals with an Arduino, you use the *analogWrite(write_value);* function. This function takes the *write_value* variable which can be from zero to 255, and uses that number to set the width of the pulses as shown in **Figure 12**. There are three terms you need to describe what is happening: **frequency, period**, and **pulse width.**

■ FIGURE 10: *Arduino Uno PWM pins.*

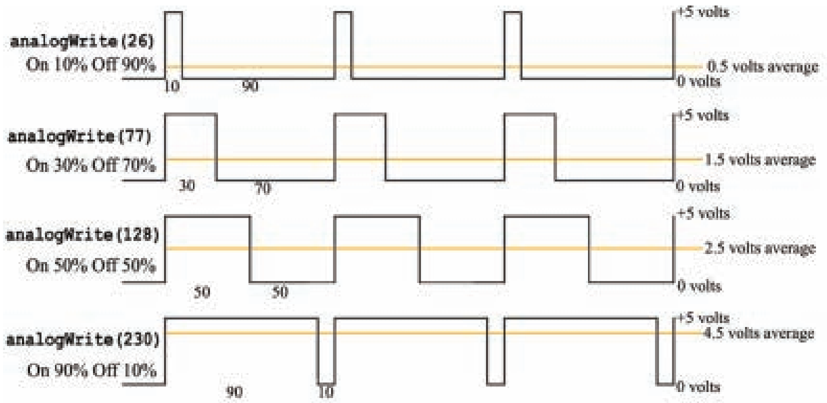
The frequency is the number of times a repeating event occurs per unit time. The period is the duration of one cycle of that repeating event. We use the term **Hz** (**Hertz**) to indicate how many times the repeating event occurs in one second. The Arduino *analogWrite* function outputs a pulse train with a frequency of about 490 Hz that equates to a period of about 2 ms.

We see in **Figure 12** that each of these pulse trains creates a constant **DC** (**Direct Current**) voltage that does not change over time. Since we can set the pulse to any of 256 widths, that means we can have 256 discrete DC voltages. If we use *analogWrite(0);*, we get a voltage of zero; if we use *analogWrite(255)*, we get the maximum voltage that (in our case) is five volts. If we use *analogWrite(127)* — half way between the two — then we get half of five volts, or 2.5 volts.

You can calculate the *write_value* to use in *analogWrite(write_value)* to give a percent of time the pulse is high by multiplying the percent times 255. Thus, for a 90% on time, you multiply 0.9x255 = 229.5. Since we are using integers, you round it up to 230. This is shown in **Figure 12** as *analogWrite(230)*.



■ FIGURE 11: *Pulse width and period.*



■ FIGURE 12: *Pulses and average voltage.*

Lab 1: Controlling LED Brightness.

Parts required:
1 Arduino
1 USB cable
1 Arduino proto shield
1 LED
1 1,000 Ω resistor

Estimated time for this lab:15 minutes

Check off when complete:
❏ Connect the LED to pin 11 as shown in **Figures 13** and **14**.

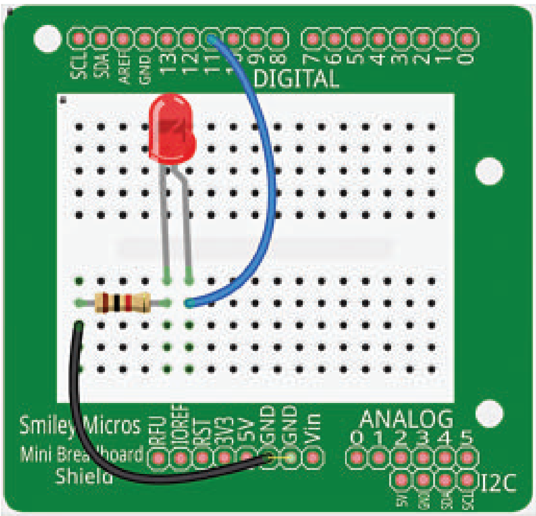


FIGURE 13: *LED to pin 11 drawing.*

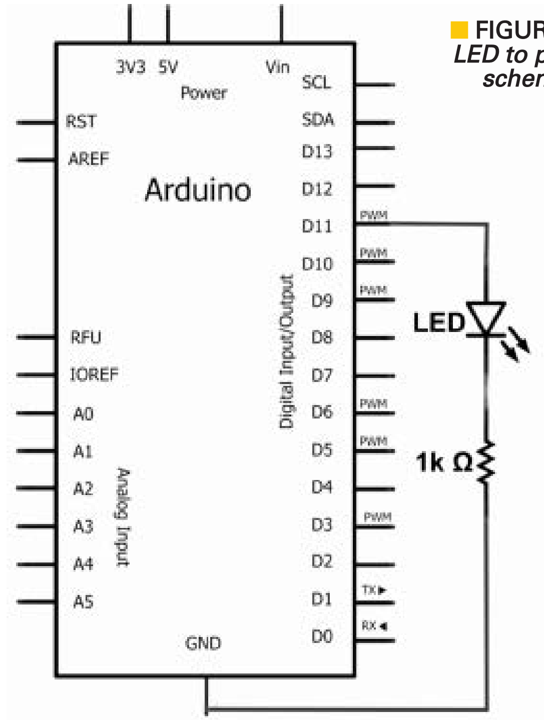


FIGURE 14: *LED to pin 11 schematic.*

❑ Open the *Set_LED_brightness* sketch, or copy the following code:

```
// Set_LED_brightness 3/11/14

int ledPin = 11; // LED connected to PWM pin 11

void setup()  {
  Serial.begin(57600);
  Serial.println("Enter 0 to 255 to set LED
brighness");
}

void loop()  {
  int brightness;

  if(Serial.available()) {
    brightness =  Serial.parseInt();
    // convert input to int

    analogWrite(ledPin, brightness);  // set PWM

    Serial.print("Brightness: ");
    // print the brightness
    Serial.println(brightness);
  }
}
```

Open the serial monitor and set several values between zero and 255 as shown in **Figure 15**.

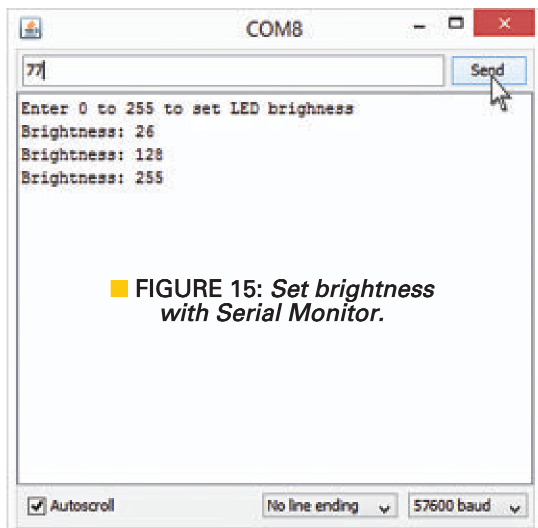


FIGURE 15: *Set brightness with Serial Monitor.*

Lab 2: Analog Output to Pulse an LED — Heartbeat LED.

In Chapter 3, Lab 2: Timing, Part 2 — Dimming the LED, we looked at using the *delay* timer to dim the LED. That technique is fairly low resolution compared to using the *analogWrite()* function. In this lab, we will use the LED setup from Lab 1 and run the Arduino example program.

However, if we use the *analogWrite()* command, the dimming is done by the timer/counter peripheral that controls the PWM on/off timimg of the pin that drives the LED. It does this in the background so that it doesn't affect the main program.

Parts required:
1 Arduino
1 USB cable
1 Arduino proto shield
1 LED
1 1,000 Ω resistor

Estimated time for this lab: 10 minutes

Check off when complete:
❑ Open the Arduino file menu and select File/Examples/Analog/Fading as shown in **Figure 16**.

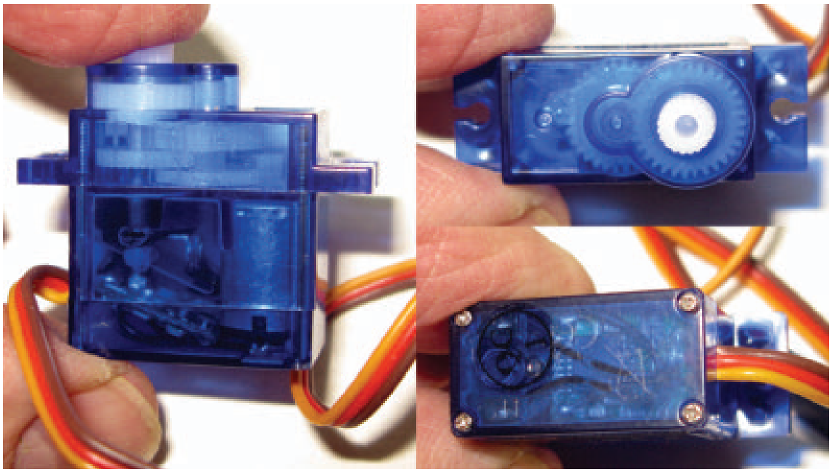


FIGURE 19: *Servo side top and bottom.*

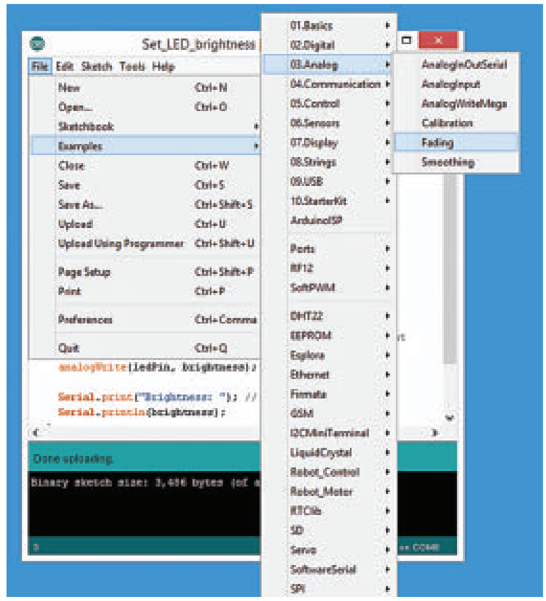


FIGURE 16: *Select the Arduino Fading example.*

- ❑ Change the *int ledPin = 9;* to *int ledPin = 11;*.
- ❑ Upload the code and observe the LED pulsing.
- ❑ Change the delay from 30 ms to 10 ms in both the fade in and fade out sections, and observe the faster pulse.

Servo Motors

When we learn about the Arduino as a controller, we learn that it controls things by looking at the environment with sensors; using what it senses, it can make things move in the environment with **actuators**. An example of an actuator is a **servomotor**. These are used in many applications in machine automation, robotics, and RC (Radio Control) models.

The word **servo** is a shortened version of **servomechanism**: a device that has built-in sensing to help control some aspect of the mechanism's function. A servomotor senses the motor actuator arm position, for example, and can vary that position over approximately half of a full turn — from 0° to 180° (angle degrees). Note that since the servo has internal sensing, the Arduino doesn't need to sense the angle position of the servo, but needs only to send it the correct PWM value to set an angle. Our servomotor does this internal angle sensing with a built-in variable resistor known as a **potentiometer** (we will learn lots more about these things next time).

The potentiometer produces a variable voltage that depends on the angle of the servomotor's rotor position. A dedicated controller in the servomotor uses that voltage to set and hold the required angular position of the actuator arm. We discussed the Arduino PWM and saw that *library* sends out pulses at a frequency of 490 Hz, which is about 2 ms per pulse. Servomotors, however, use a 10 ms to 20 ms period; within that pulse, they require an on time between 1 ms and 2 ms to control the angular position as shown in **Figure 17**. Fortunately, the Arduino has a servo library that sets the correct pulse width and

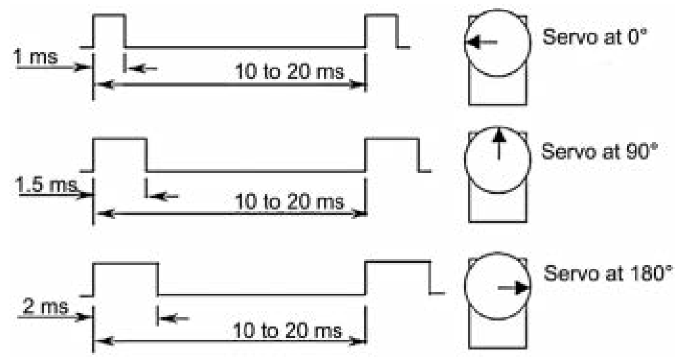


FIGURE 17: *Servo pulse and angle.*

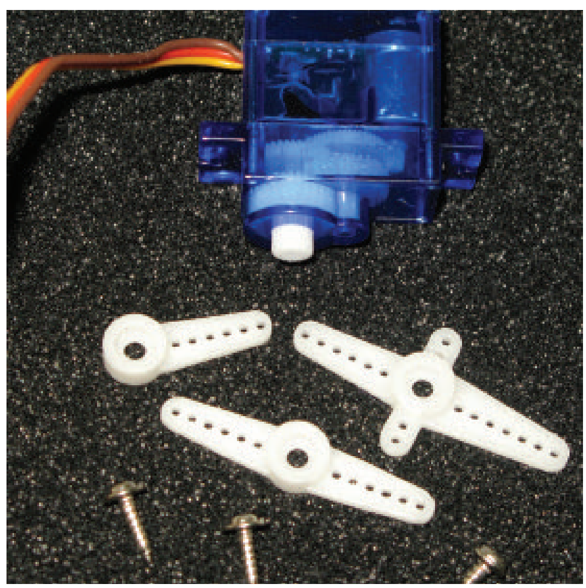


FIGURE 18: *Servo horns and screws.*

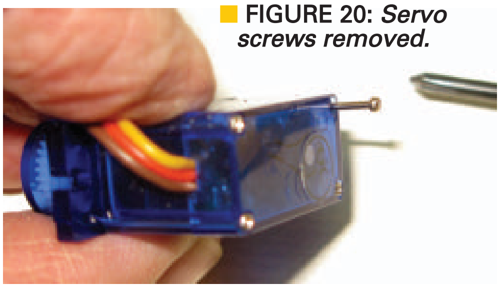


FIGURE 20: *Servo screws removed.*

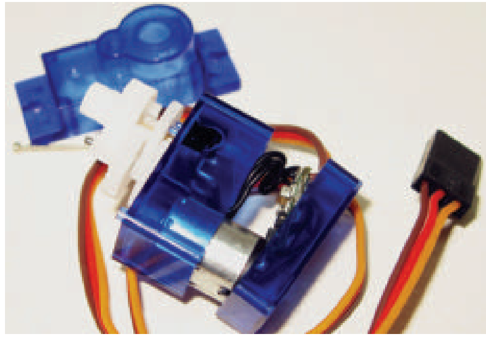


FIGURE 21: *Servo top and bottom removed.*

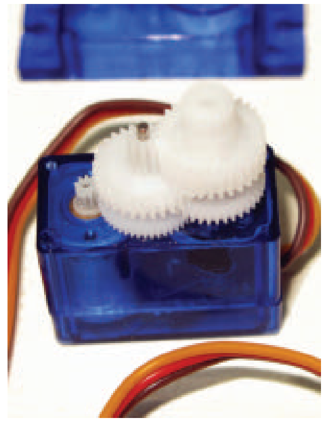


FIGURE 22: *Servo top off.*

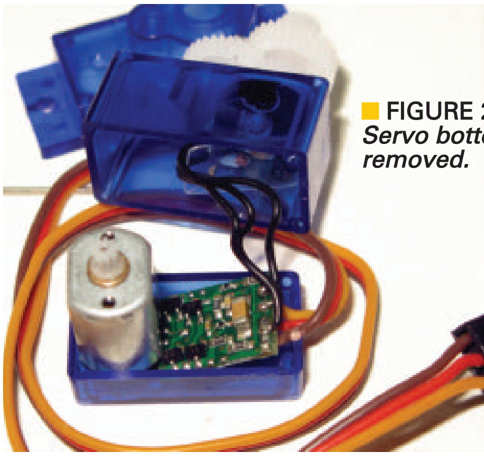


FIGURE 24: *Servo bottom removed.*

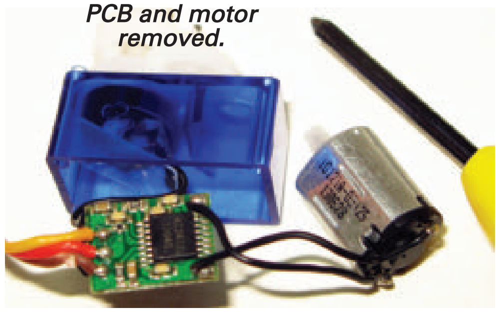


FIGURE 25: *Servo PCB and motor removed.*

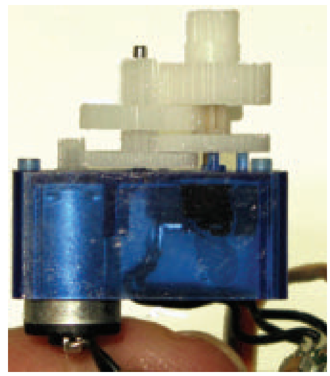


FIGURE 23: *Servo side with top and bottom off.*

period so that we can use an angle as a parameter to set the angle of the servomotor. We will learn to do this in the following labs.

What is Inside Our Servomotor?

The following materials will save you from tearing apart your servomotor to see how it works. Not that you shouldn't do that, but I'd suggest waiting until you've done all the labs in this chapter and in Chapter 6. You know ... just in case you break something.

Figure 18 shows the servomotor in the Arduino 101 Projects kit that is available from the *Nuts & Volts* Webstore. The three white pieces are called horns, and are attached to the servomotor shaft with a small screw. The large screws are used to attach the servomotor to a frame or base. The horns are used to provide attachment points for rods that are used in RC models to control the position of things such as rudders and elevators on model aircraft.

In **Figure 19**, you can see the internals of the servomotor through the transparent plastic with the label removed. We took off the bottom screws as shown in **Figure 20** which allows us to remove both the top and bottom of the servomotor case as shown in **Figure 21**.

Figures 22 and

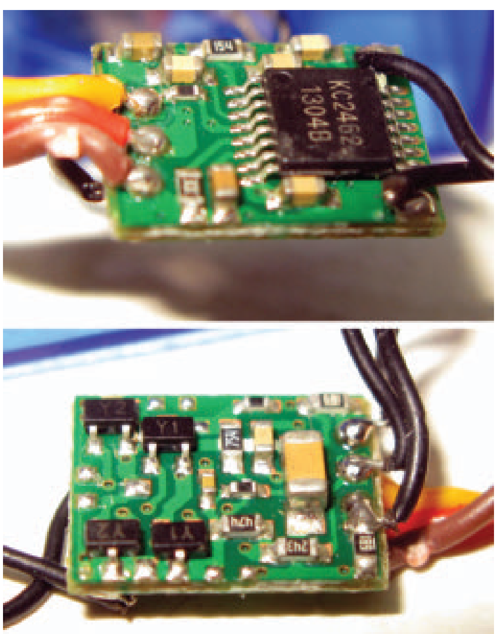


FIGURE 26: *Servo PCB top and bottom.*

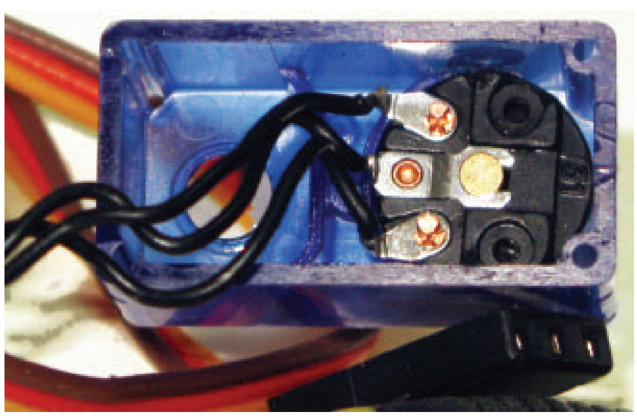


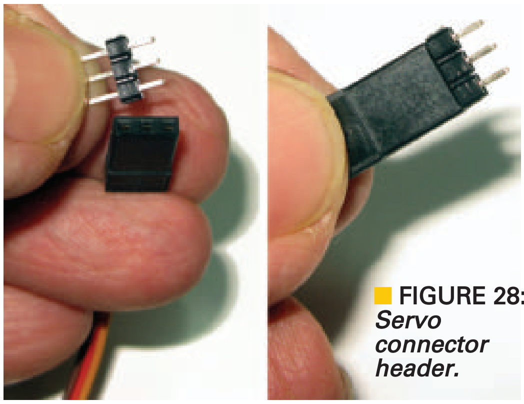
FIGURE 27: *Servo potentiometer.*

23 show details of the gear system. In **Figures 24** and **25**, we see the motor and controller board removed from the bottom of the servomotor. **Figure 26** shows the bottom and top of the printed circuit board (PCB) that contains components used to control the position of the servomotor shaft. **Figure 27** shows the variable resistor (potentiometer) that is connected by the gears to the motor and the external shaft used by the horns. The potentiometer turns when the shaft turns, and the resistance varies with the degree of the turn. This resistance is measured by the controller on the PCB and is used to control the angular position of the shaft.

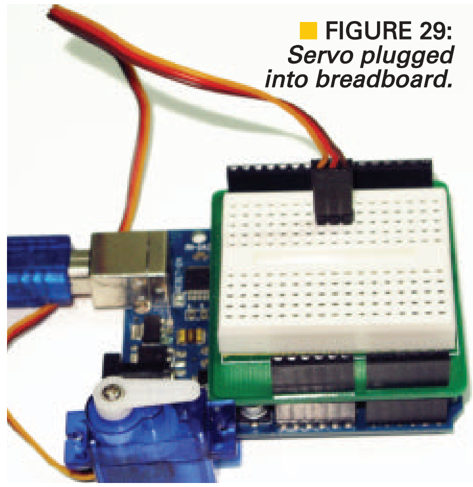
Lab 3: Connecting and Testing the Servomotor.

Parts required:
1 Arduino
1 USB cable
1 Arduino proto shield
1 Three-pin header
1 Mini servomotor

Estimated time for this lab:
15 minutes



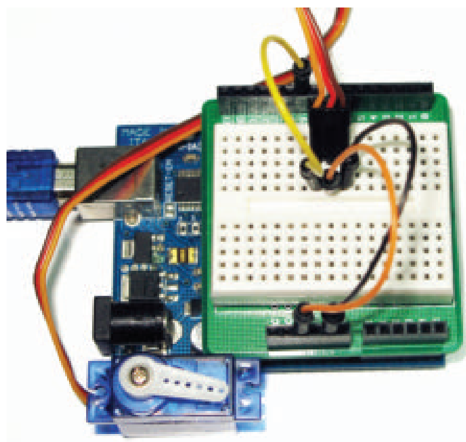
■ **FIGURE 28:** ***Servo connector header.***

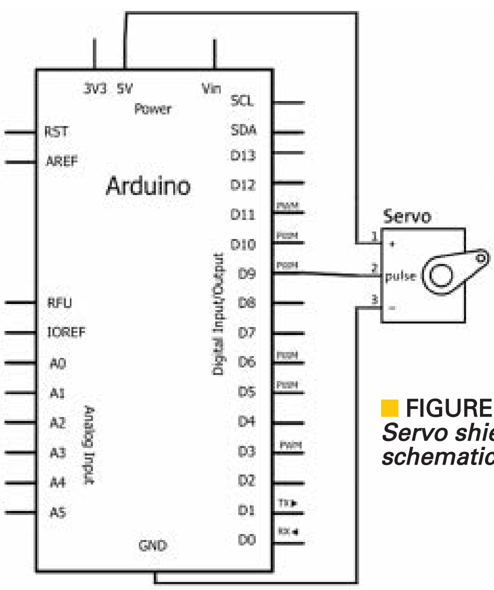


■ **FIGURE 29:** ***Servo plugged into breadboard.***

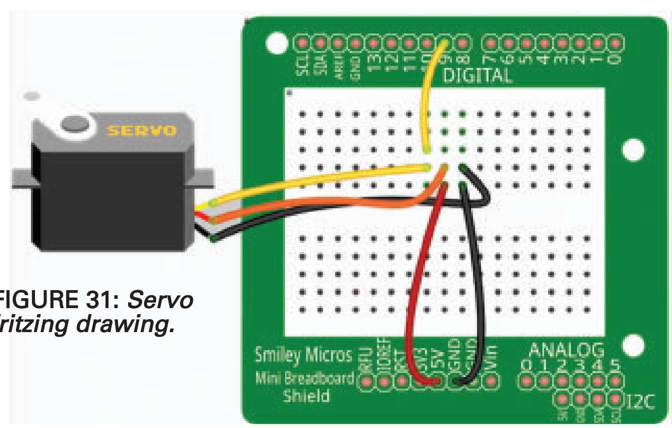
Check off when complete:
❏ Place the long legs of the three-pin header into the mini servo connector as shown in **Figure 28**.
❏ Plug the three-pin header into the breadboard as shown in **Figure 29**. These pins are short so you'll need to exercise care to keep the header plugged into the breadboard.
❏ Connect a jumper wire from the position of the mini servo yellow wire to pin 9 on the Arduino header as shown in **Figures 30**, **31**, and **32**.
❏ Connect a jumper wire from the position of the mini servo orange wire to the 5V pin on the Arduino header as shown in **Figures 30**, **31**, and **32**.
❏ Connect a jumper wire from the position of the mini servo black wire to the GND pin on the Arduino header as shown in **Figures 30**, **31**, and **32**.
❏ Open the Arduino IDE and select the *File/Examples/Servo/Sweep* example sketch.
❏ Upload the sketch.
❏ The servo should begin to sweep back and forth 180°.

■ **FIGURE 30:** ***Servo wired to Arduino.***





■ **FIGURE 32:** ***Servo shield schematic.***



■ **FIGURE 31:** ***Servo Fritzing drawing.***

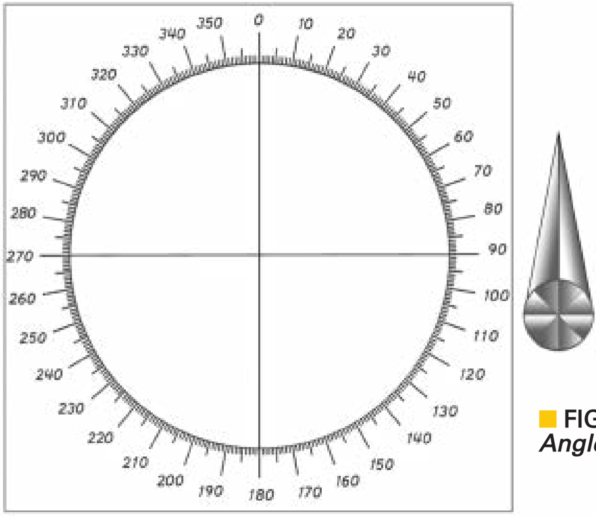


FIGURE 33: *Angle's dial.*

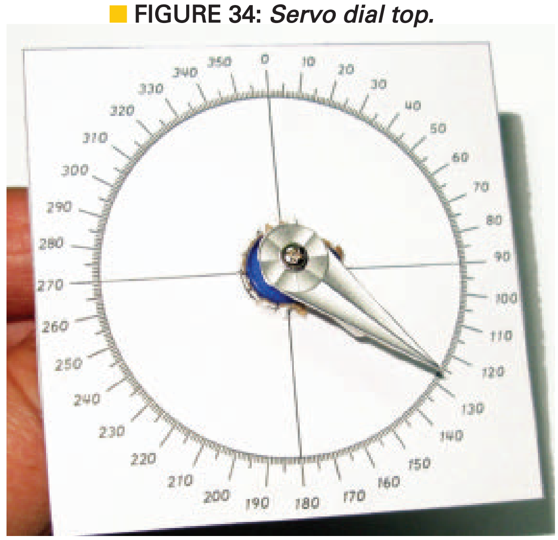


FIGURE 34: *Servo dial top.*

Lab 4: Setting an Angle on a Servomotor.

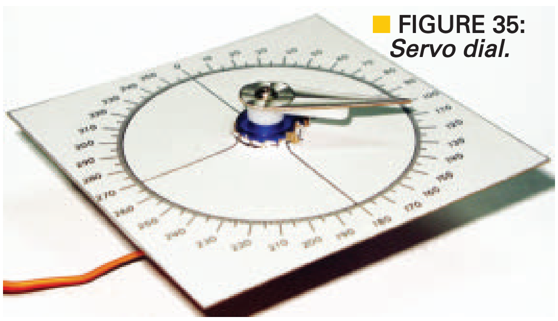


FIGURE 35: *Servo dial.*

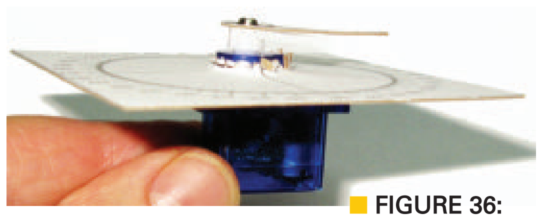


FIGURE 36: *Servo dial side.*

Parts required:
1 Arduino
1 USB cable
1 Arduino proto shield
1 Three-pin header
1 Mini servomotor
1 Printout of the servo angle and pointer image shown in **Figure 33**
1 Piece of cardboard (cereal box) to accommodate the compass and dial
1 Double-sided sticky tape
1 Pair of scissors

Estimated time for this lab: 30 minutes

Check off when complete:
- ❑ Print the angle's dial and pointer shown in **Figure 33**. [Note this image can be found at the article link.]
- ❑ Place strips of double-sided sticky tape on the back of the angle's dial and pointer printout.
- ❑ Place it on the cardboard (from a cereal box, for example) and carefully smooth it down.
- ❑ Cut out the angle's dial and pointer.
- ❑ Look at **Figures 34**, **35**, **36**, and **37** to get a feel for how to cut out the holes to mount the compass and the dial.
- ❑ Use scissors to make a tiny hole in the center of the compass, then clip out a shape like the top of the mini servo — a larger circle at the center and a smaller circle to the side. This should match the top of the mini servo in shape, but be about 1/8" smaller.
- ❑ Clip 1/8" notches around the shape so that you can force the mini servo into the hole; have the clipped edges squeezed tight to the mini servo.
- ❑ Use scissors again to make a small hole in the base of the dial, then use the smaller of the screws shown in **Figure 34** to attach the dial to the mini servo.
- ❑ Open the *A101_5_Servo_Angle.ino* program [at the article link].
- ❑ Notice that the servo angle and the compass angle are in opposite directions, so the code is modified to show the angle entered into the serial monitor as the angle the servo points to:

```
// Servo Angle
// 3/7/14 Joe Pardue

#include <Servo.h>

Servo myservo;  // create servo object to
                // control a servo

int pos = 0;  // variable to store the servo
              // position
String readString;

void setup()
{
  myservo.attach(9);   // attaches the servo on
                       // pin 9 to the servo
                       // object

  // initialize the serial communication:
  Serial.begin(57600);
  Serial.flush();
  Serial.println("Servo Angle 1.0");
```

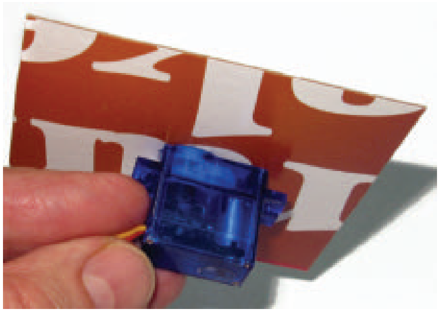


FIGURE 37: *Servo dial bottom.*

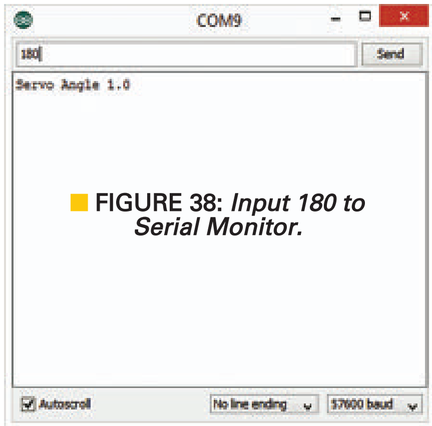


FIGURE 38: *Input 180 to Serial Monitor.*

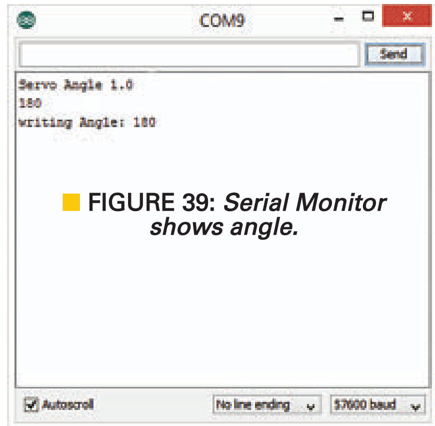


FIGURE 39: *Serial Monitor shows angle.*

```
}

void loop()
{
  byte angle;

  while (Serial.available()) {
    char c = Serial.read();
    //gets one byte from serial buffer
    readString += c;
    //makes the string readString
    delay(2);  //slow to allow buffer to fill
               //with next character
  }

  if (readString.length() >0) {
    Serial.println(readString);
    //so you can see the captured string
    int n = readString.toInt();
    //convert readString into a number

    // The compass has the angles reversed from
    // the servo angles so we convert the input
    // angle to fit the compass angle
    n = 180-n;

    Serial.print("writing Angle: ");
    Serial.println(180-n); // show angle sent
    myservo.write(n); // use converted angle

    readString=""; //empty for next input
  }
}
```

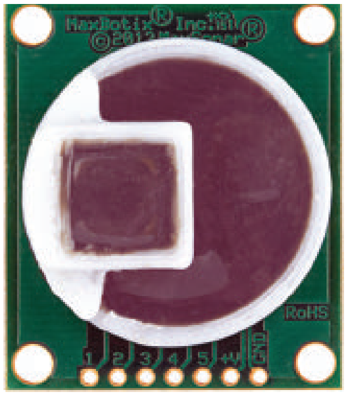
❏ Upload the program.
❏ Open the serial terminal and send it a value of 180 as shown in **Figures 38** and **39**.
❏ After the servo has set the angle, loosen the pointer screw and move the point of the tip to point at 180 on the angle dial.
❏ Now, send various angles such as 45, 90, 135, etc., to test that the mini servo angle is being set correctly.

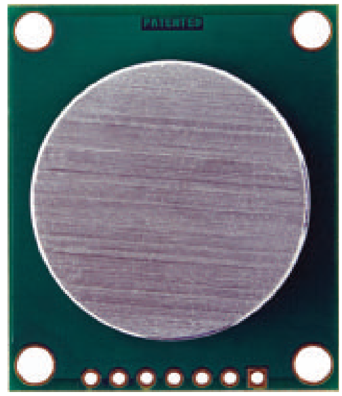
That's a wrap for this chapter. Next month, we will learn about analog input and potentiometers, and we will also turn our Arduino into a voltmeter. **NV**

NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

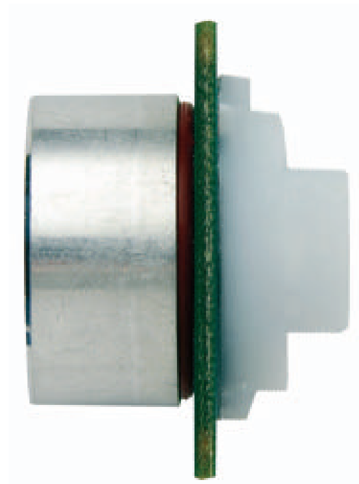
ULTRA-COMPACT ULTRASONIC SENSOR SERIES





Maxbotix's popular sensor is now available in a new design which is physically shorter than any of their current outdoor sensors, allowing easy integration into applications. The new UCXL-MaxSonar-WR series sensors are flexible OEM-customizable products intended to be integrated into a customer's system with a horn, or designed for flush mounting into an existing housing. These rugged high performance sensors are individually calibrated to provide quality.

Mounting design recommendations are provided through Maxbotix's 3D CAD models (available in multiple formats) to facilitate the design process. The UCXL sensors are RoHS and IP67 compliant with proper mounting design, and the sensor layout offers four conveniently placed mounting holes for design flexibility.

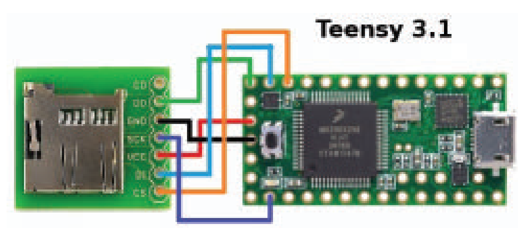


The UCXL-MaxSonar-WR comes with easy to use outputs and standard pin configurations of previous MaxSonar products. There are three standard sensor outputs of RS-232 serial (TTL output available upon request), analog voltage, and pulse width. These sensors feature 1 cm resolution, an operational temperature range from -40°C to +70°C (-40°F to +160°F), real time automatic calibration (voltage, humidity, ambient noise), 200,000+ hours mean time between failure, an operational voltage range from 3.0V-5.5V, with a low 3.4 mA average current requirement. These sensors are also RoHS Compliant and CE Compliant.

For more information, contact:
Maxbotix
Web: **www.maxbotix.com**

BASIC COMPILER FOR TEENSY MICROCONTROLLER

An ARM BASIC compiler from Coridium is now available for use on PJRC's Teensy 3.0 and 3.1 ARM controllers. For US$5, hobbyists and students can download a copy for themselves. This is the full compiler with string, IEEE 754 floating point, and SDcard file system support. The ARM BASIC compiler is ideal for:



- Reseachers designing prototypes
- Factory automation and data collection
- Students learning how to program embedded controllers
- DIYers exploring robotics, wireless, and sensors

The Teensy from **PJRC.com** is a complete USB-based microcontroller development system in a very small footprint. With support for three UARTs, I²C, SPI, and one-wire devices, Coridium's BASIC makes it easy to integrate a variety of peripherals into a project. For commercial users, licensing is available, as well as customization that can include Ethernet web, ftp, mail, and UDP support.

For more information, contact:
Coridium
Web: **www.coridium.us**

DIY ELECTRONIC AMP KITS

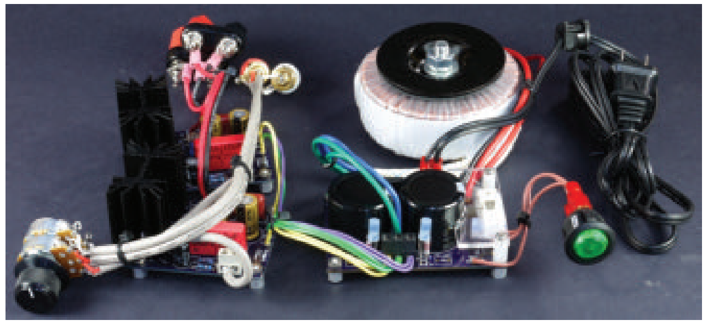
Boxed Kit Amps introduces two new DIY electronics kits: the SoloG Pocket Guitar Amp Kit and the Case Free Gobo Stereo Amp Kit. Both kits boast easy-to-follow step-by-step instructions.

The pocket guitar amp retails for US$25 and is a learn-to-solder kit or a quick fun build for a more experienced maker. It's suitable for any age qualified to handle a soldering iron. There are only through-hole components and the only wiring is to the battery holder. This guitar amp will fit in a back pocket and can be used with either headphones or speakers, or plug it into a regular amp to add another groovy effect.

The SoloG is Boxed Kit's take on the venerable "Smokey" amp. It uses a genuine JRC386D op-amp just like the original. This is buffered by a JFET-based emitter follower to get clean powerful tube-like sound like the famous "Ruby" amp, but the similarity ends there.

Instead of a redundant "volume" control or any sound shaping, Boxed takes advantage of the guitar's volume and tone knobs. A red LED on the pocket amp acts as a clipping indicator for the gain control, and it's backed by a couple of Si diodes (it wouldn't be an audio circuit without at least one 1N4148). Using the LED, musicians can adjust the gain so there's no clipping, or can crank it.

The Case Free Gobo Stereo Amp Kit is a Class AB chip amp kit that lowers the bar and the cost for anyone wanting to build their own high-quality stereo amp. Assemble the Gobo's high-end audio-grade components, build a case, and show off the warm, retro sounding custom analog-based chip amp design.



The Case Free Gobo kit lists for US$170 and includes:

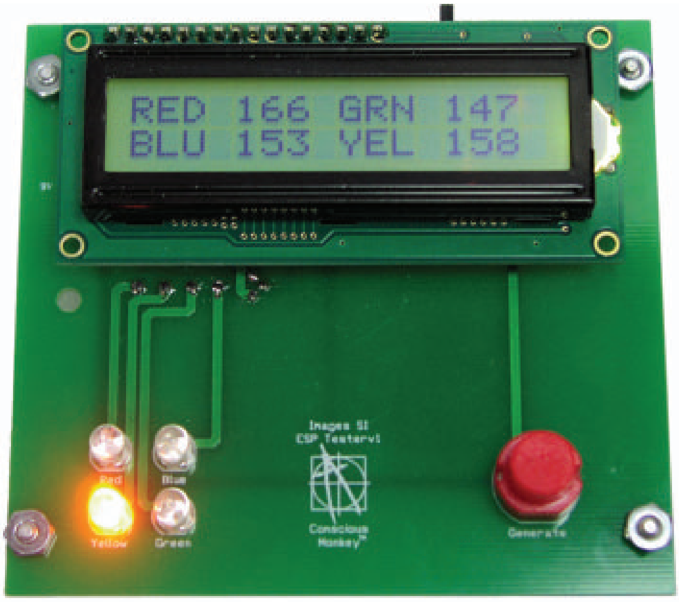
- Easy-to-follow instructions crafted with the beginner in mind, rich with high-quality photos.
- The popular op-amp LM1875 which is easy to use, stable, and provides excellent performance.
- Hand-routed amp boards designed with the shortest possible paths and smooth lines.
- A power board with oversized capacities and flexible options.
- Professional-grade toroidal audio transformer engineered specifically for audio applications — cool running, quiet, and efficient-boasting minimal interference from stray signals.
- High-quality audio grade capacitors and resistors.
- Bling in the form of an anodized aluminum knob, stainless steel hardware, and gold plated connectors.

For more information, contact:
Boxed Kit Amps
Web: **www.boxedkitamps.com**

MANUAL ESP/PSI TESTER

The manual ESP/PSI tester kit now available from Images SI is an interesting human-input random number generator (RNG) device that can be used to check ESP/PSI, and the mind's ability to influence (PK) and forecast random events.

The ESP tester utilizes a momentary contact switch that a user presses to generate true random numbers. The PIC16F88 in the ESP tester is spinning at approximately

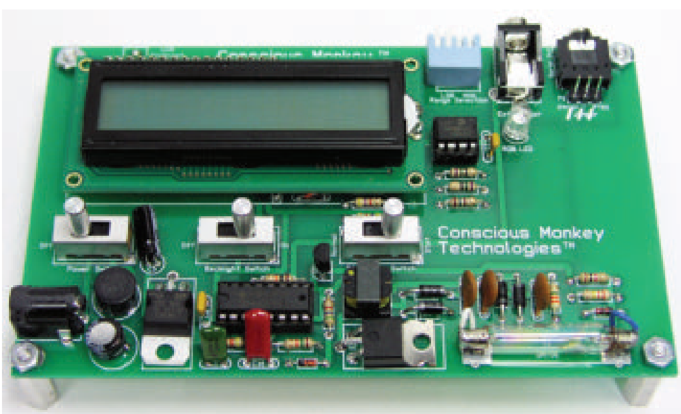


1,000,000 numbers a second. The instant the microcontroller sees the user press the momentary contact switch, it stops spinning and the current number is selected as the random number.

The RNG's output is connected to four different colored LEDs. Depending on the random number generated, one of the four colored LEDs is lit. The 16x2 LCD keeps a running tally of the random numbers generated for different PSI tests. While this setup might appear trivial, it is not, and may be used to accurately test for different aspects of ESP. Random event generators (REG) are similar to RNG, and sometimes the terms are used interchangeably.

The included manual explains how to perform different ESP and PSI experiments. The PIC RNG source code is included in the manual. The suggested retail price is US$39.95 for the ESP tester kit which requires soldering.

STAND-ALONE RANDOM NUMBER GENERATOR



Also available from Images SI is the RNG-01 kit which is a lab quality instrument. It uses the immutable randomness of radioactivity decay to generate random numbers. Quantum mechanics states that the nuclear decay of atoms are fundamentally random and cannot be predicted.

This device has a mini Geiger counter that detects background radiation. The detection of a radioactive particle is a random event used to initiate the generation of a random number. The output of the built-in Geiger counter is monitored by a PIC which rotates numbers inside a register at approximately 1,000,000 a second. When a radioactive particle (random event) is detected, the microcontroller stops the rotation of the numbers, reads the current number in the register, and produces a random number.

The RNG-01 will produce approximately one to three random numbers every minute from background radiation. These true random numbers are useful for data encryption (cryptography), statistical mechanics, probability, gaming, neural networks and disorder systems, PSI and ESP testing, micro PK experiments, etc.

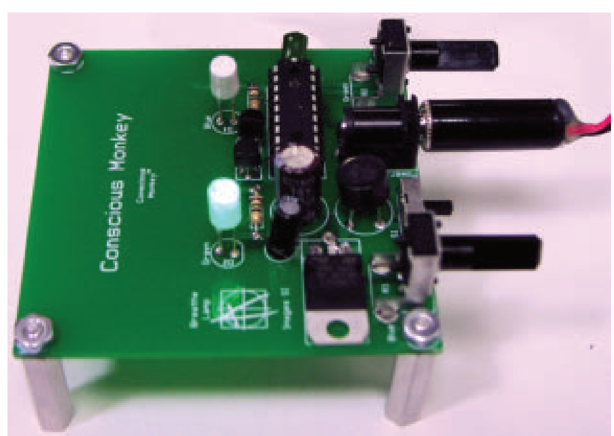
The RNG-01 allows selection of the following random number ranges: 1-2, 1-4, 1-8, 1-16, 1-32, 1-64, and 1-128. It displays random numbers generated on an LCD and outputs the random number via a serial port. When range selection is 1-4, the LCD can also display the running tally of the four numbers generated to verify their distribution. The RNG-01 has a four color LED output.

The RNG-01 is currently available as a kit which contains a printed circuit board, LCD screen, preprogrammed PIC, and all the necessary components for construction.

The retail price is US$119.38 for the RNG-01 kit (which requires soldering.)

BREATH PACER LAMP

There are many reported benefits of slow focused breathing, such as reduced blood pressure and stress and greater focus and concentration.



Breath pacer lamps are designed to help the user take a few relaxed breaths throughout the day. As with all breath techniques, it is important to progress gradually and comfortably.

Images' new breath pacer lamp has two controls on the back. One control adjusts the length of one breath: an inhale and exhale. The time period can be varied from approximately six seconds to 20 seconds.

The other control adjusts the ratio of the inhale to the exhale. The ratio starts at 1:1 and can go up to approximately 1:9. Most strive for a 1:2 ratio, with the exhaling breath being two times longer than the inhale breath.

Set the breath pacer lamp to a comfortable inhale and exhale breathing routine. Do this by first paying attention to the normal breathing pattern. Then, adjust the lamp's green and blue LEDs to follow that pattern. Place the lamp where it is visible so it can be used throughout the day.

Images' breath pacer lamp is available as a kit (soldering required) which includes a printed circuit board, pre-programmed microcontroller, and all the electronic components needed for assembly. Suggested retail price is US$43.45.

For more information, contact:
Images SI, Inc.
Web: **www.imagesco.com**

72 Watt Constant Voltage/ Constant Current POWER SUPPLY

By Allen Ripingill
aripingill@comcast.net

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/ may2014_Ripingill.**

FIGURE 1. CV/CI power supply.

I have built many power supplies in the past. All of these power supplies used a transformer to connect the mains power to the rectifier and regulator circuits. In all cases, a linear regulator was used to provide the final output voltage. Adding adjustable current limiting was too difficult. The input voltage to the regulator dropped at full load because of the output resistance of the transformer.

I discovered a Constant Voltage/Constant Current (CV/CI) battery charging module that can be repurposed into a power supply capable of delivering up to 30V at up to 5A in a size that would astound you. It measures 2" x 1" at one inch deep. It also includes a three-digit DVM and a three-digit ammeter. It includes three LEDs for current limiting, for battery charging, and for charge complete. It has two trim pots: one to adjust the voltage and another to adjust the current. It costs less than $12 on eBay, including a heatsink and shipping. This heatsink might cost about a $1.50 more than the average price but it is well worth it in this application. The finished power supply is in **Figure 1**.

The Problem

Some of these Chinese bargain modules have hidden problems that we will fix. For example, it has two voltage regulators that exceed worst-case limits and it uses a ground side current monitoring resistor, meaning that the input ground and the output ground have a current monitoring resistor in between them. So, if they are shorted together, the ammeter and current limiter will not work. This also means that adding a second module will require another 32V DC input power supply.

The first problem is the voltage regulators for +5V and +3V. The 5V regulator (for the current monitor) uses a 78L05 and draws about 5.25 mA. The input voltage is rated at 30V absolute maximum; we are using 32V. I have already blown one 78L05 regulator in the module.

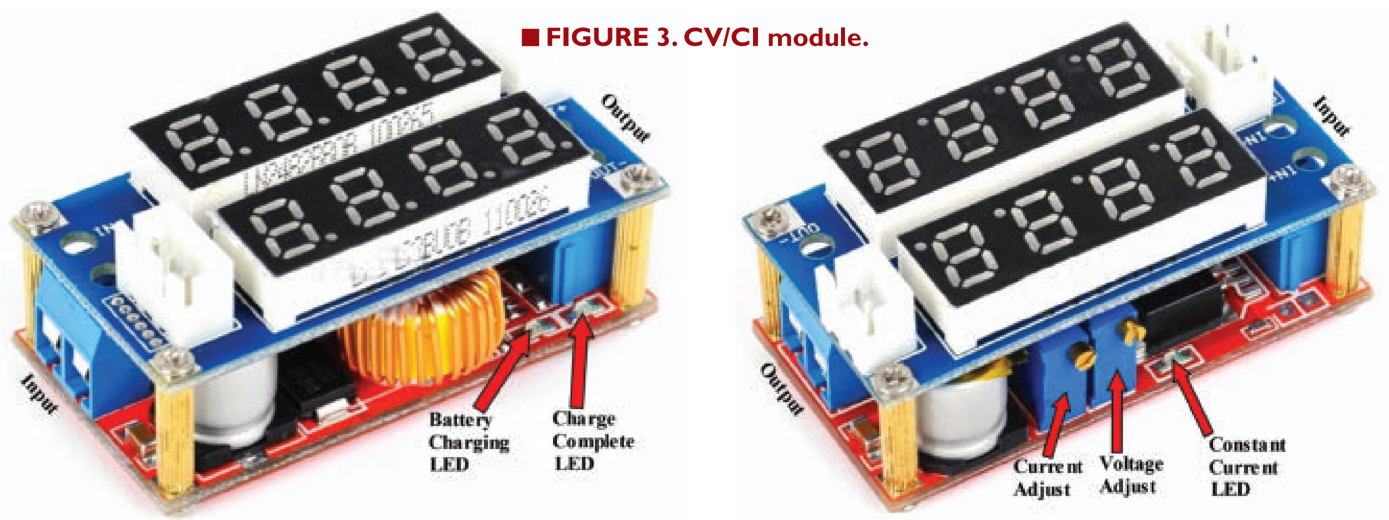
The display board (schematic not shown) uses an HT7130-1 3V regulator and draws about 9.5 mA. There are a couple of problems here. First is the input voltage is rated at 26V absolute maximum (we are running it at 32V). Second, the power dissipation is rated at 250 mW and we are running it at over 300 mW. We will fix these later.

For full dynamic range, the module requires a 32V DC input power source. The module that I selected was a used HP 0957-2093 AC power adapter.

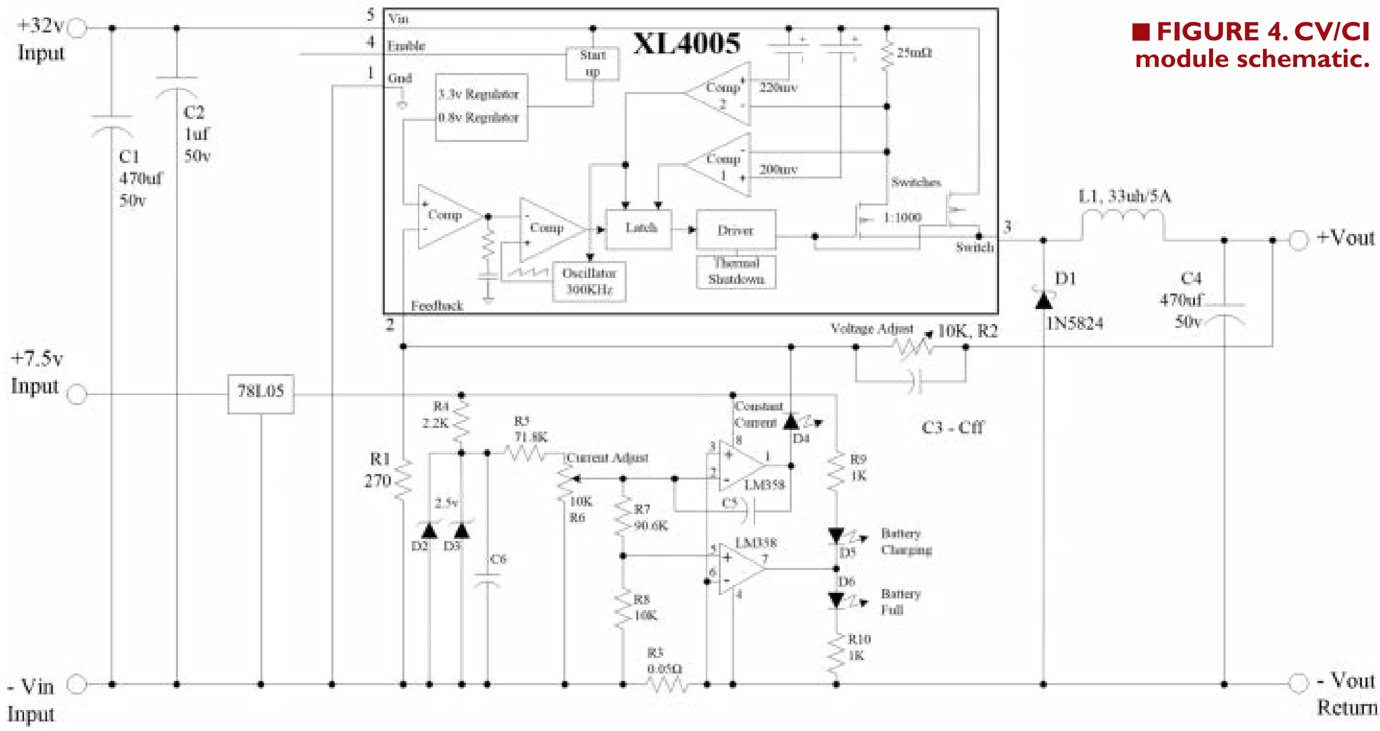
■ **FIGURE 2. HP 0957-2093 AC power adapter.**

Theory of Operation

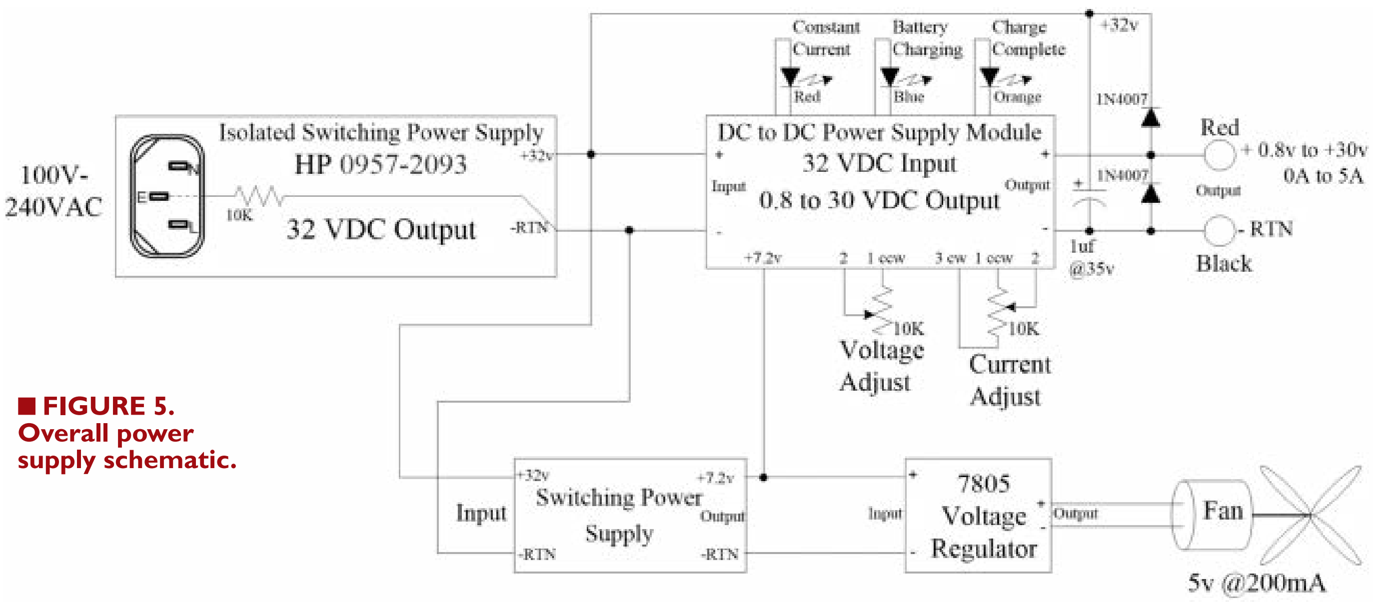
Let's begin with the HP 0957-2093 AC-to-DC power adapter. Its input voltage is 100V-240V AC (50-60 Hz); refer to **Figure 2**. The output voltage is 32V DC, with a current output capability of 2.5A DC, or 80 watts. **Figure 3** shows two views of the CV/CI module. It is a step-down module with an output voltage of 0.8V-30V at a rated current of up to 5A. It incorporates a red LED panel meter with a three-digit voltmeter and current meter display.



■ **FIGURE 3. CV/CI module.**



■ FIGURE 4. CV/CI module schematic.



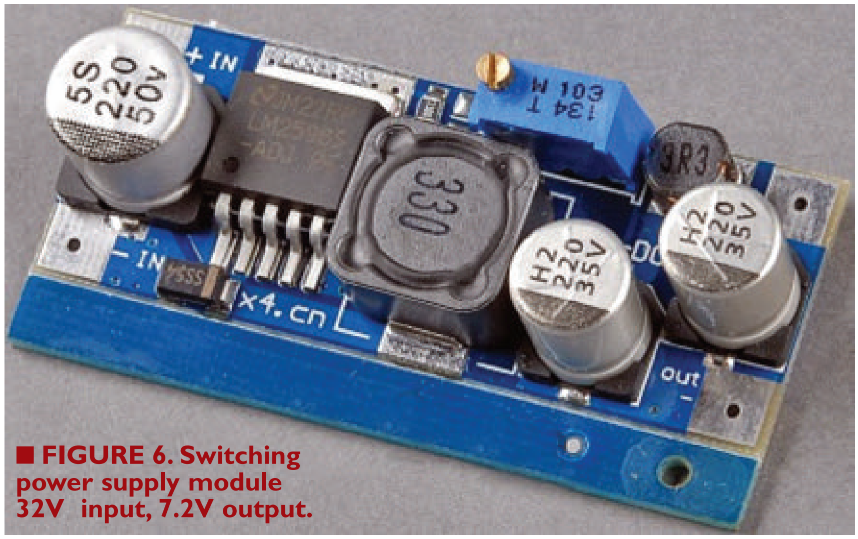
■ FIGURE 5. Overall power supply schematic.

Figure 4 shows the schematic for the CV/CI module; it does not include the display board.

I have shown the 7.5V input as a separate input. There will be more on this later. The schematic consists of two separate sections: the upper section, which is a switching regulator utilizing an XL4005; and the lower section, which is a current monitor and set point for the current limiting. This module uses a conventional switching regulator IC with a voltage adjust feedback resistor R2 to set the output voltage point. There are two FETs inside the XL4005 that switch at up to a 300 kHz rate, and drive L1 and C4 to establish the output voltage.

The lower section uses R3 to convert the output current to a voltage. The digital ammeter in the display shows the output current which is proportional to the voltage across R3. The LM358 compares the voltage drop across R3 to a set point established by variable resistor R6. R6 derives a precision reference voltage from zeners D2 and D3. When the voltage drop across R3 exceeds the set point voltage established by R6, the output of the LM358 at

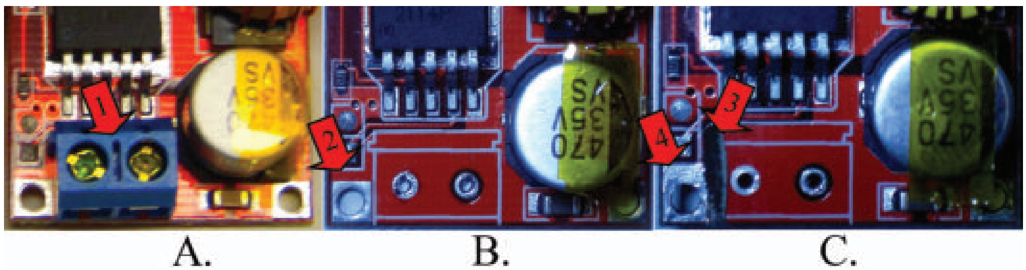
pin 1 goes high until the LED D4 forward biases. This voltage starts rising and the voltage at the feedback point pin 2 of the XL4005 causes the switching regulator to begin to shut down. The side benefit of forward biasing D4 is that it begins to light and shows that the switching regulator is current limiting. The second comparator of the LM358 uses a set point that is 1/10 of the current limiter value established by R6. This comparator is used when charging rechargeable batteries. It indicates when the rechargeable battery reaches a trickle current by lighting the charge complete LED.



■ **FIGURE 6. Switching power supply module 32V input, 7.2V output.**

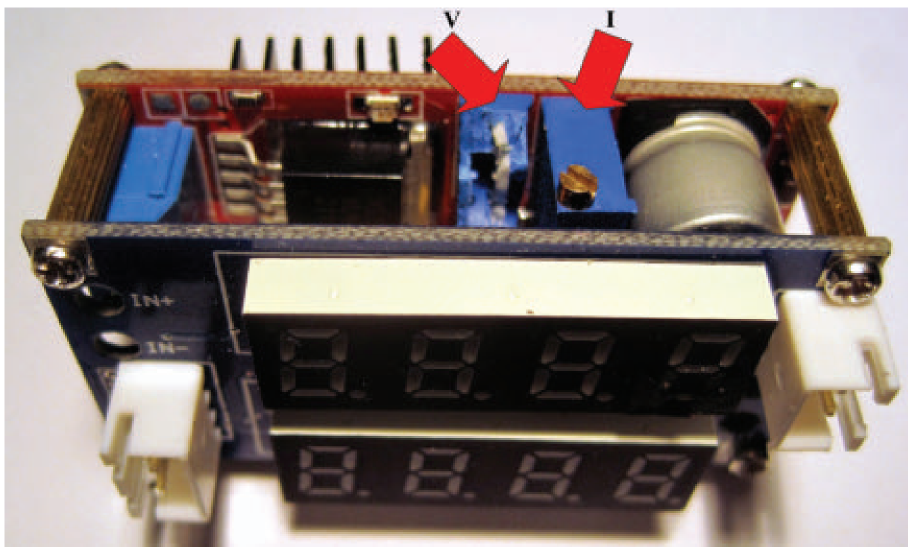
Putting It All Together

To solve the problem of over-voltage to the +5V and +3V regulators, we must modify the CV/CI module; refer to **Figure 7**. To do this, we must first separate the CV/CI module switching regulator from the display board. Carefully remove the blue input terminal block from the switching regulator board shown in **Figure 7** (A "1"). We must then scrape off the red solder mask shown in B "2" and cut the trace shown in C "3." Then, the scraped area from step B "2" must be tinned as shown in C "4." It might be wise at this point to use solder wick to remove any excess solder. After reassembly of the module, we will be soldering a red wire to this tinned area.



■ **FIGURE 7. CV/CI module PCB modification.**

The last step is to put the blue input terminal block back on; be careful to assemble it in the right direction. Remove the two trim pots. I think the best way to do this is to cut them apart with a pair of dikes until the wires are flush with the circuit board. If you try to save the pots at this point, you may damage the circuit board which is far more valuable than the trim pots: see **Figure 8**.

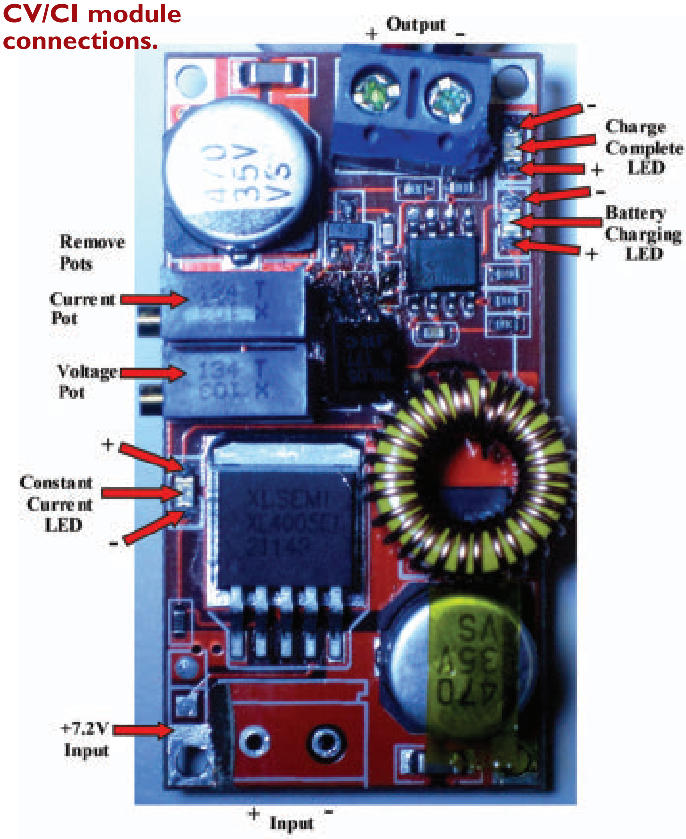


■ **FIGURE 8. CV/CI module trim pot modification.**

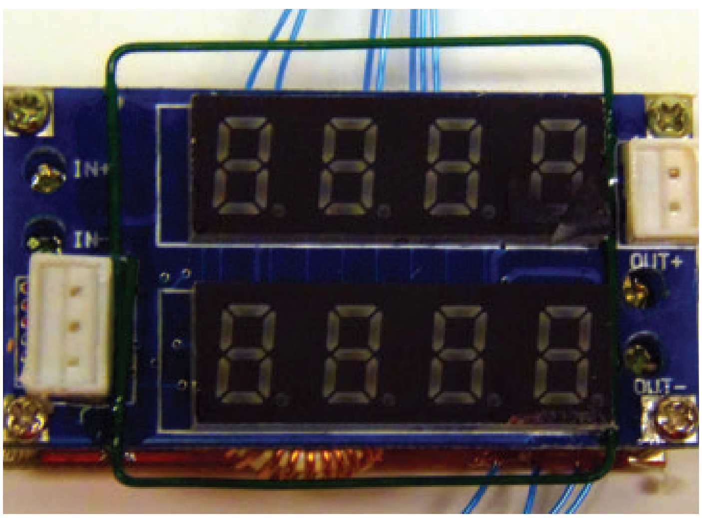
Using solder wick, remove the charge complete LED, the battery charging LED, and the constant current LED. Now, connect a wirewrap wire to the plus and minus solder pads of each of the three LEDs, along with the current pot and the voltage pot. I like to use hot melt glue to prevent the wires from breaking the pads off and to secure them to the circuit board. Connect the two boards back together with the standoffs. Solder a red wire to the +7.2V input; this will be connected later. Take a look at **Figure 9**.

For mounting the module, I took some insulated steel floral wire as shown in **Figures 10** and **11**. I used five minute epoxy to tack the wire down. Next, I prepared my front panel. First, I printed the front panel artwork, then I

■ FIGURE 9. CV/CI module connections.



■ FIGURE 10. CV/CI module mounting wires.



taped it to the front panel aluminum. I center-punched all locations of the holes, drilled the holes, and used a nibbler to cut out the square for the display slightly larger than the blue display border. Finally, I carefully epoxied the locking wire for the module to the front panel.

To prepare the front panel, print the front panel artwork (available at the article link) on nice quality photo paper. Using an X-Acto™ knife, cut out the square for the display leaving the blue border. Put a clear cover sheet over your artwork using Avery Clear Full-Sheet Labels (#8665), for example, then apply an Aleene's Tacky Double-Stick Sheet to the rear of the front panel artwork.

Prepare a red transparency the size of the aluminum display opening. Carefully trim the outside of the front panel artwork, remove the backing paper, and apply it to the aluminum front panel. The red transparency can now be placed in the hole of the aluminum front panel and it will stick to the backside of the display artwork. Before mounting the display, we must change the voltage indicator from a "U" to a "V." To do this, place two small triangles of black electrical tape on the display as shown in **Figure 12**.

Now for final assembly. Mount all of the front panel components including the CV/CI module as shown in **Figure 13**. Mount all of the components inside the box that includes the fan, 5V regulator (for the fan), 7.2V regulator, and power entry connector, also shown in **Figure 13**.

Wire the pots to the CV/CI module as in **Figure 14**. The two outside solder pads of the voltage pot are connected together, so only two wires to the pot are needed: one to CCW (pin 1) and one to the tap (pin 2). The wire wound potentiometer pinouts are as follows: 1 = CCW; 2 = Slider; and 3 = CW. When wiring the constant current LED, be very careful with the + and - as a wiring error here will cause death of the switching regulator XL4005 when the power supply is supposed to go into current limiting. I would recommend that you use your VOM to check the wiring.

Do you want your masterpiece to last? Broken wires are a major cause of early mortality of your home-built circuit. To prevent these wires from breaking, add a little hot melt glue over the ends of the wire to cover the insulation to the bare solder joint; refer again to **Figure 14**. This also needs to be done at the other end where the wire connects to terminals or components such as the LEDs.

Testing

Start out with the power off by turning the voltage and amperage knob completely counter-clockwise; then, go up about two full turns. Before powering up your power supply, check the wiring. Turn the pot counter-clockwise on the LM2596S DC-DC buck converter module.

You are now ready to power-up the module. Plug in your HP 0957-2093 supply and set the trim pot of the LM2596S DC-DC buck converter for an output voltage of 7.2V. The CV/CI power supply should now be displaying a voltage of about 6.5V and 0A. You should be able to set the voltage between .8V and 30V.

Do not short out your power supply to set the current limit with the power on as this could blow your XL4005 from the sparks. (I think Nikola Tesla and Guglielmo

Marconi showed that sparks could produce a great deal of RF with their radio transmitter experiments. This can be death to the XL4005 output FETs.) I learned this after blowing two modules. Two diodes have been added to help prevent this problem.

To set your current limiting: 1) Shut off the power supply; 2) Place a short across the red to black output terminals; 3) Turn on your power supply; 4) Set the current limit; 5) Turn off the power supply; and 6) Remove the short. Your power supply is now ready to use. Set the desired output voltage and the current will automatically limit at the value set in step 4 discussed above.

Operation

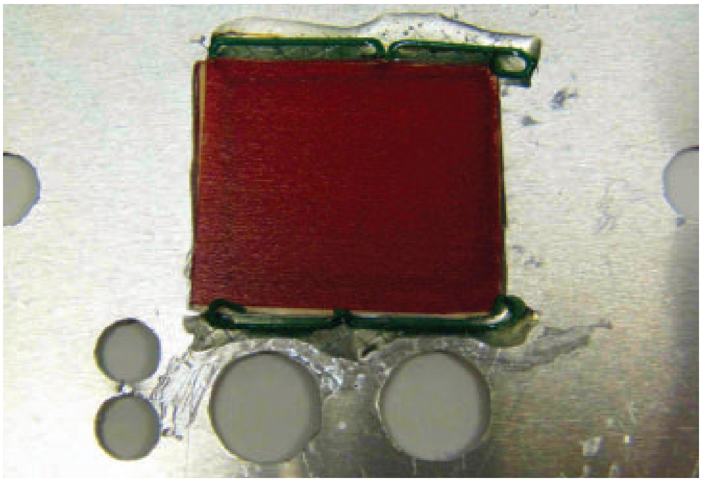
Using the voltage adjust knob, you should be able to set the voltage between .8V and 30V. To set the current limit, follow the procedure in the paragraph above.

Why Use a CV/CI Power Supply?

Have you ever blown up a breadboard or electronic project when you turned on the power supply? If you haven't, you will. Burning resistors give off a very distinctive smell. Plastic transistors, ICs, and electrolytic capacitors can literally explode. So, how do you prevent this catastrophic event? We start by estimating the current your masterpiece will draw. Then, we multiply that value by 1.5 to 2, and set the constant current to that value. Next, set the output voltage. Now, you may connect the circuit under test to your CV/CI power supply and turn it on. If the power supply immediately goes into current limit, you may have a wire hooked up wrong or the electrolytic capacitor in backwards.

Double-check your wiring, power supply connections, and electrolytic capacitors. If all that checks out okay, it could be your circuit's current estimate was too low.

The next step requires safety glasses. I know this from experience as it took about a month for the welt on my forehead to heal when an electrolytic capacitor (wired in backwards) blew up. Now, turn the power supply on and with a maximum current in mind (maybe x4), SLOWLY turn up the current limit and watch for the power supply voltage to stabilize to your preset value or to have smoke appear (whichever comes first).



■ FIGURE 11. CV/CI module mounting.

■ FIGURE 12. Voltage display tape to make a "V."

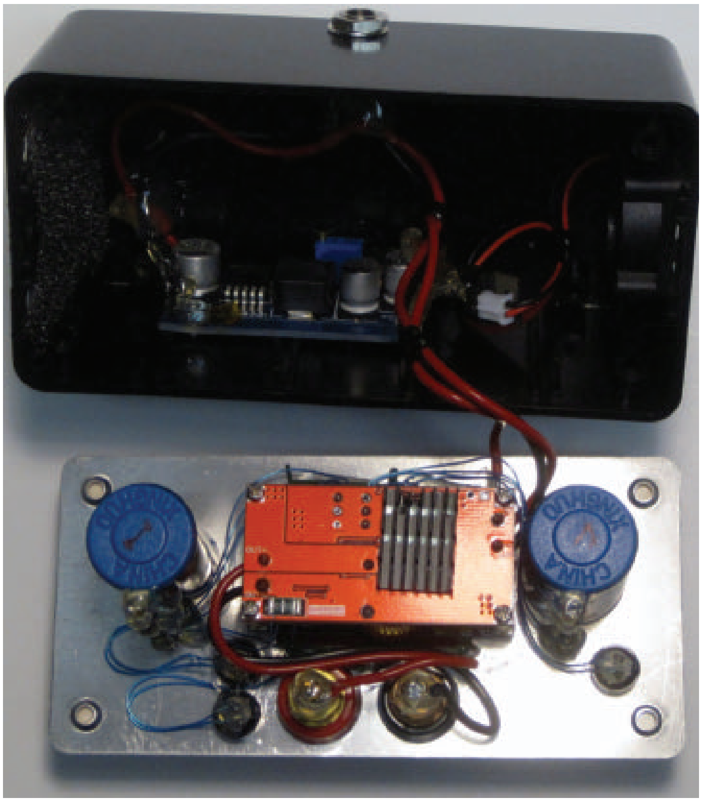
Battery Charging — Using a CV/CI Power Supply

A constant voltage/constant current power supply can be used as a battery charger for rechargeable batteries. Overcharging or overheating batteries can be very dangerous and cause them to explode. A CV/CI power supply can charge with a preset maximum constant current until it approaches the preset constant voltage setting, at

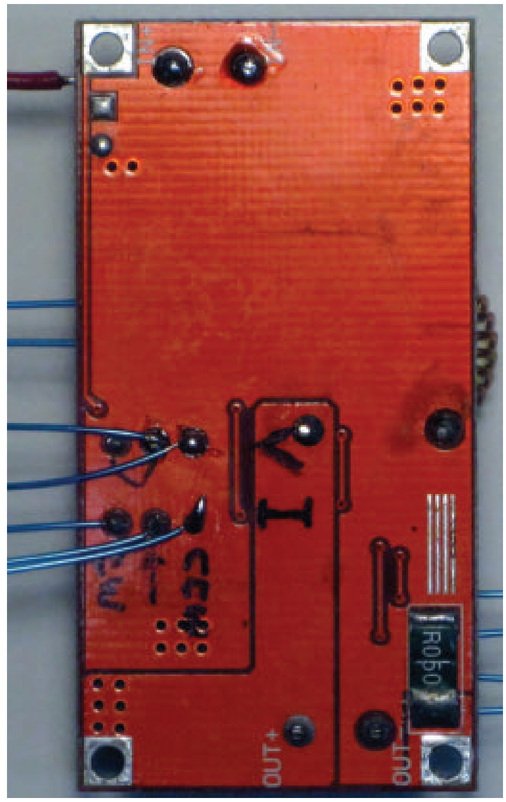
PARTS LIST

HP 0957-2093 AC Power Adapter; ~$10
5A Constant Current/Voltage Module with Voltmeter and Ammeter (USA seller ***nyplatform***, includes a heatsink); ~$11.50
LM2596S DC-DC Buck Converter Module 32V input, Adjustable Output Voltage; <$2
2x 10K ohm Multi-Turn Wire Wound Potentiometers WXD3-13; $4.50
Project Enclosure (5" x 2"), Model 270-1803 **Radio Shack.com**; $5.50
7805 - 5V Regulator; <$1
2x 1N4007 Brushless DC Fan 5V @ 0.12A; <$4
Red and Black Binding Posts; ~$1
Red/Blue/Orange LEDs; $3 @ 100 pcs
LED Holder/Chrome Metal; <$2 @ 10 pcs
2x Knobs (Plastic); <$5.50 @ 5 pcs
Lockable 5.5 mm x 2.1 mm DC Power Plug plus Socket; <$2.50

NOTES:
1) All products were purchased from eBay unless otherwise listed. Check out your usual favorite suppliers instead where possible.
2) eBay prices typically include shipping.



■ **FIGURE 13. Overall power supply wiring complete.**



■ **FIGURE 14. Wiring for the voltage pot and current pot.**

which time the current automatically reduces to a trickle charge. CV/CI power supplies do not have temperature monitoring to prevent battery explosions, so do your homework on the type of battery you'll be charging to determine the maximum voltage and current. When charging batteries, do not leave them since the charge termination is not always automatic.

You must first set the final charged voltage based on the number of cells in series for the type of cells you are charging. Then, set the maximum current to charge the cells.

When determining the charge time, a reference "C" is used for the rate of charge current. A "C/1" would be the Ah (amp hour) value to fully charge or discharge a battery in one hour. We must supply more energy to the battery than its actual Ah capacity to account for energy loss during charging. A trickle charge rate might be C/10. One might think that it would take 10 hours, but 14 hours is necessary because of the energy lost charging the battery – usually by heat.

These numbers can vary from manufacturer to manufacturer	NiMH to 1.41V mAh	NiCd to 1.41V mAh
AA	1,300	700
AA	2,450	1,000
AAA	850	300
AAA	1,000	
C	4,500	3,000
D	10,000	5,000

Table 1.

If we had a 100 mAh battery, then C/10 would be 100/10 or 10 mA for 14 hours. To rapid-charge this battery (C/1), we would have to charge 100 mA for about one hour and 15 minutes. If we try to increase the charge rate faster than C/1, we run the risk of overcharging and damage to the battery that results from heat generated in the battery. Caution must be taken when charging C and D batteries as many manufacturers will disguise an AA battery by packaging it in a C or D battery case. This affects the C/1 value.

To wrap things up, here are some tips for specific batteries:

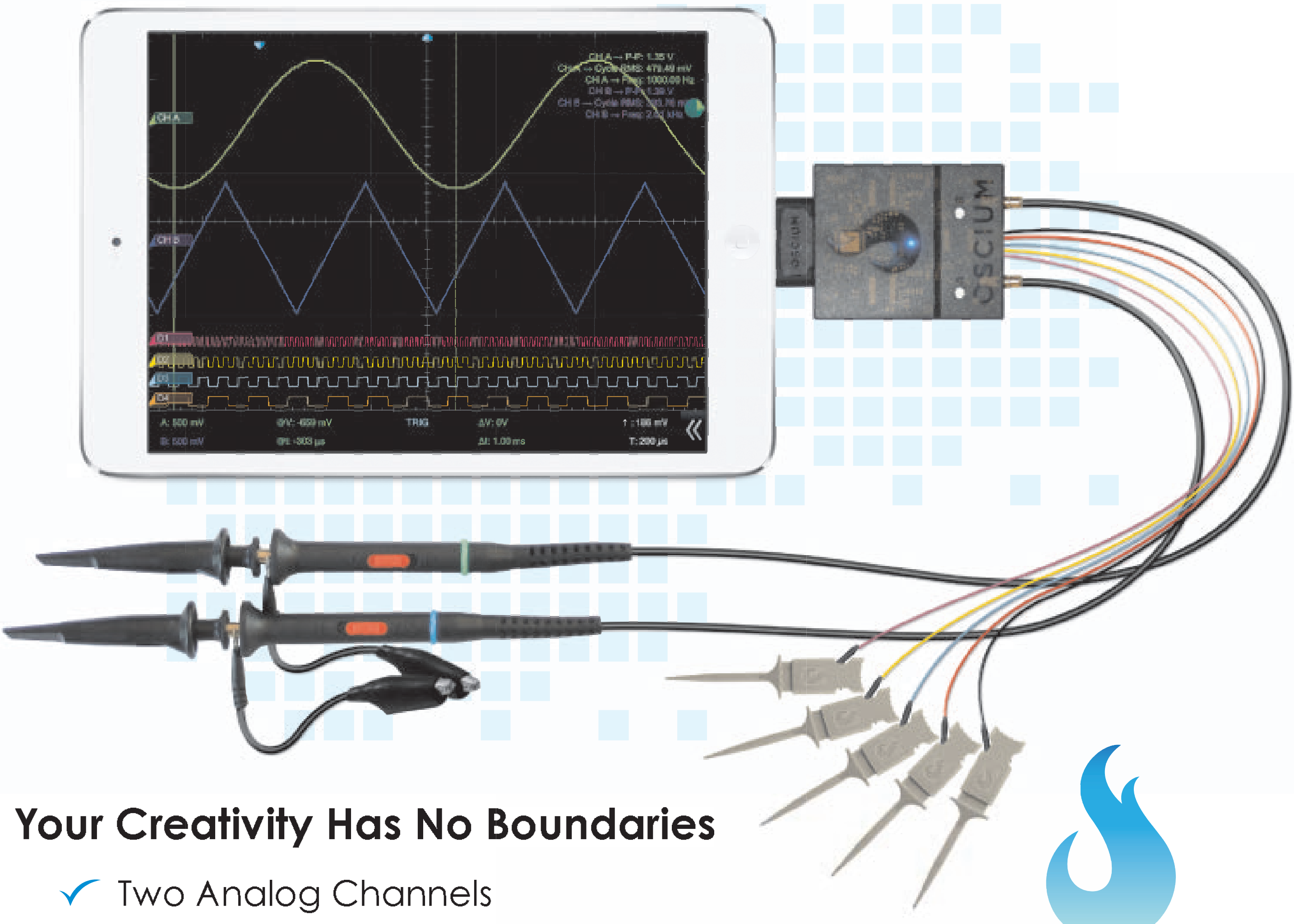
Nickel–Metal Hydride (NiMH) and Nickel–Cadmium (Ni–Cd) batteries: Charge to 1.41 volts per cell and use a timer to prevent overcharging. Do not continue past 13 to 15 hours (with C/10). A moderate charge rate might be C/2 for 2.5 to 3 hours. Refer to **Table 1**.

Sealed Lead–Acid Batteries: The volts per cell vary with temperature and manufacturer. For example, 2.6V at 0°C, 2.45V at 25°C, and 2.3V at 50°C. The C/1 or Ah is all over the place with each type of battery and manufacturer. You have to do your homework on this one, too.

Charging **lithium-ion, lithium-polymer, and lithium-iron phosphate batteries** can be dangerous and requires close supervision. Be sure you research this thoroughly before proceeding. **NV**

Calling All Inventors & Makers...

Transform iPad, iPhone, & iPod Into An Oscilloscope



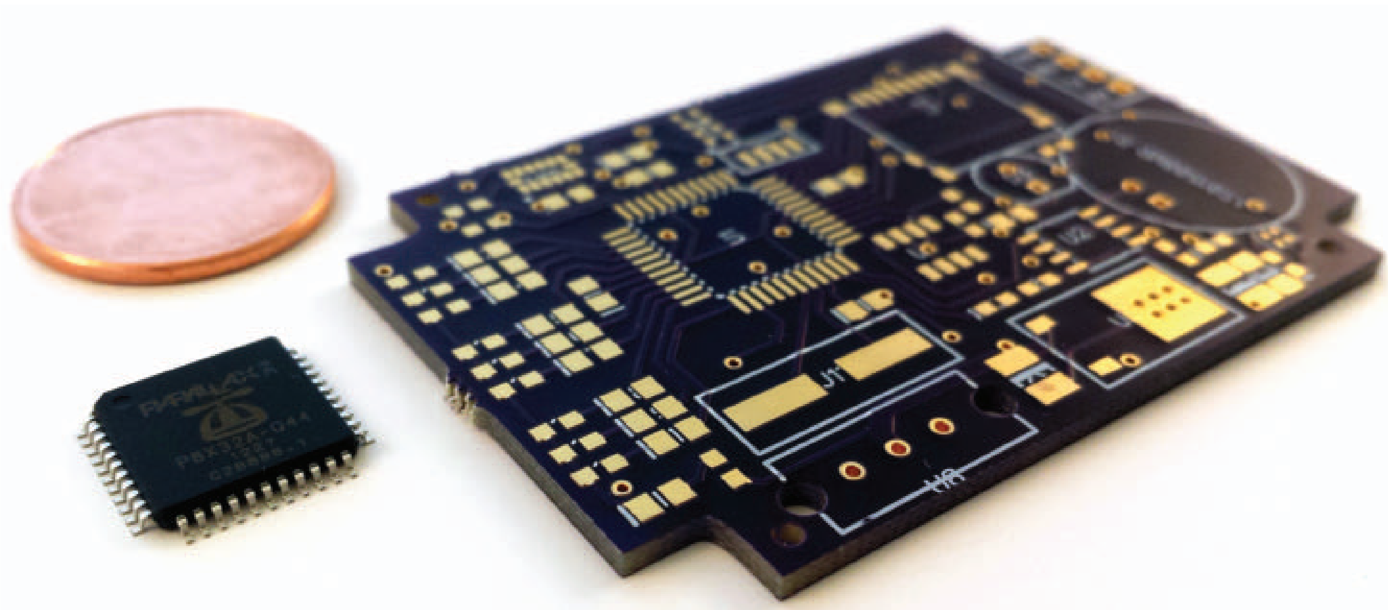
Your Creativity Has No Boundaries

- ✓ Two Analog Channels
- ✓ Ultra-Portable (Pocket sized)
- ✓ 50 MSPS
- ✓ Native Lightning Connector

OSCIUM
oscium.com

	iMSO-204L
Channels	2 Analog + 4 Digital
Max Sample Rate	50 MS/s
Bandwidth	5 MHz
Sample Depth	1000 pts
Horizontal Sensitivity	200ns/div-10s/div
iOS App	Download on the App Store

Do-It-Yourself Surface-Mount



By David Dorhout

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2014_Dorhout**.

"Any sufficiently advanced technology is indistinguishable from magic." - *Arthur C. Clarke*

Electronics is a wonderful adventure. At first, everything is covered with arcane symbols, but then you begin to decipher the symbols and crack the code. You blink your first LED. You continue gaining skills and advancing. So, what will you create next? Your first SM printed circuit board, of course!

From Breadboards to Printed Circuit Boards (PCBs)

Typically, most people are introduced to the breadboard during their first tutorial in electronics and use it to build their first electrical circuit. The components have long leads that are easy to insert into the breadboard and are called through hole (TH) or dual in-line package (DIP) components. For a time, pretty much every electrical component that was sold could be used with a breadboard for a temporary setup or soldered together using a perforated circuit board for a permanent solution.

You could buy the latest technology in a TH or DIP configuration and use it in your project. However, as technologies advanced and demand for more compact designs increased, parts were made smaller and the TH and DIP configurations were dropped, leaving the part only available in a surface-mount (SM) configuration (**Figure 1**).

Surface-mount components have been a great advance and allow devices to be built smaller and smaller. The only drawback is that you have to use an adapter board while prototyping with a breadboard. This isn't a major problem with many companies already selling SM sensors or components in a breadboard-friendly format, with the

Printed Circuit Boards

Check the video at www.nutsvolts.com and see how the professionals at **Parallax** apply solder paste, place components with "pick-n-place," reflow the solder in their massive oven, and solder TH components with a "solder volcano."

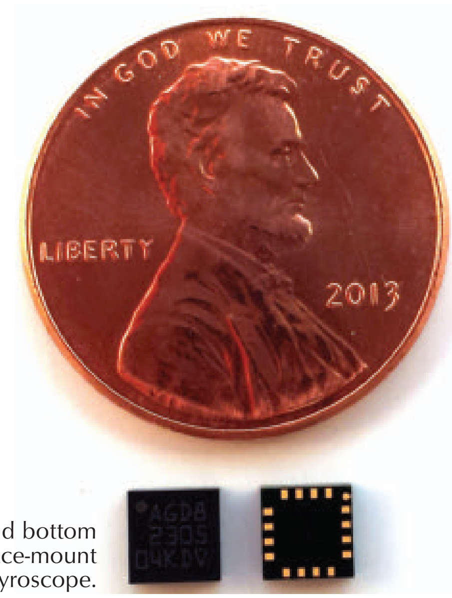


Figure 1. Top and bottom of a surface-mount three-axis gyroscope.

added bonus that much of the component's supporting circuitry (resistors, capacitors, etc.) is included. It's also possible to buy an adapter board and solder the SM component to it yourself. Several companies (like SchmartBoard) specialize in adapter boards of all sizes and pitches (lead spacing).

Now, you could use an SM component on an adapter board for your final configuration, but wouldn't the best practice be to just create a PCB and solder everything together without adapters? Creating your own PCB might sound daunting, but the rewards are more than worth the effort.

Getting Started With Making Your Own PCBs

There is a dizzying array of PCB configurations and ways to make them. This article will walk you through the general process of:

1. Selecting and using a computer aided drafting program (CAD) to design your circuit.
2. Ordering boards from a board house of your choice.
3. Ordering the solder paste stencil at a company of your choice.
4. Applying the solder paste and placing the SM components.
5. Soldering the board using a toaster oven.

By the end of this article, you should be able to build on your previous experience creating circuits with breadboards and know the basics of making your own PCBs. We're not going to be able to go into all of the details — such as how to use the CAD software — because you might want to use a different program, and each program has an excellent tutorial.

A word about safety: Please be careful! Read and follow all safety directions. Always wear and use appropriate safety equipment. I personally wear safety glasses and rubber gloves whenever I'm working with lead-based solder paste and will only solder in a well-ventilated area to avoid the fumes. If you're unsure, please consult the manufacturer or board house before beginning. You alone are responsible for your safety.

Designing Your Circuit Board

There are many different CAD programs that you can use for designing your circuit board layout. Sometimes they are abbreviated as eCAD to differentiate them from a mechanical CAD (mCAD) program that might be used to design a 3D part.

A quick Internet search will give you a list of numerous programs — free or available for purchase. I would recommend that you "try before you buy" and make sure that the software package allows you to create patterns, components, and export the Gerber/NC Drill files. Gerber files are the (nearly) universal file format that circuit board houses use to create boards. They're analogous to the ubiquitous .stl file format that mCAD programs export for use in CNC milling, or the slicer program for a 3D printer.

The pattern and component editors allow you to create the pad spacing and shape. You can use the component's datasheet in the event that the part isn't in a particular program's parts library.

I personally use an eCAD program called DipTrace that was recommended to me. (I used it to create the screenshots in this article.) This article won't be going into how to use programs or best design practices, so make sure

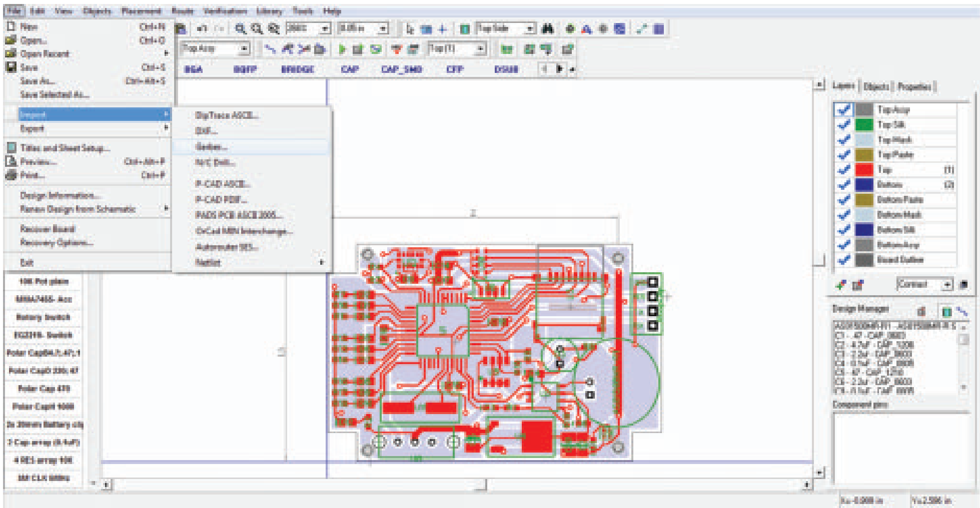


Figure 2. Selecting to export the Gerber files with a board layout in the background.

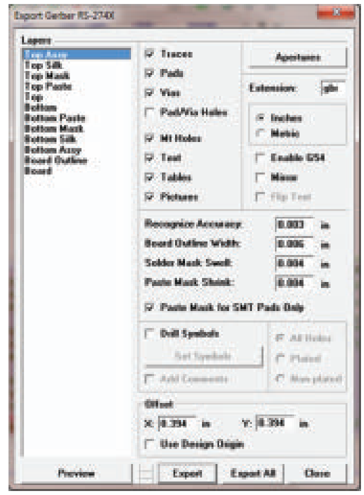


Figure 3. Gerber Export, main window.

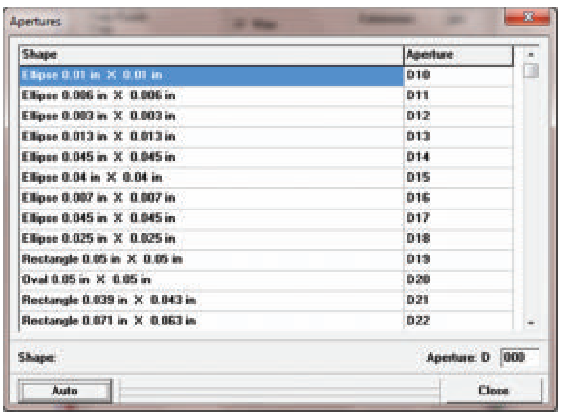


Figure 4. Aperture window.

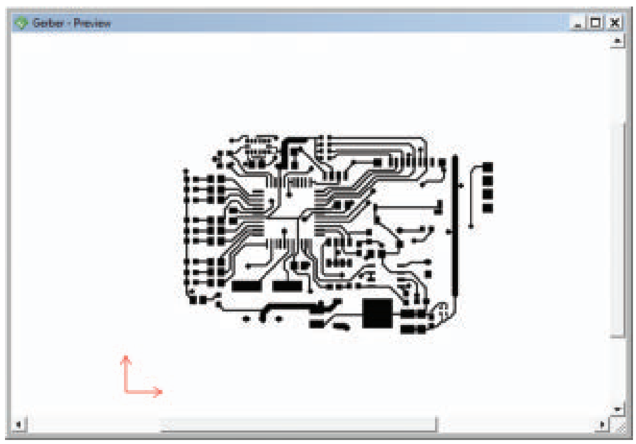


Figure 5. Top copper layer preview.

you go through the tutorial that's included with whatever software you choose.

Exporting Files

Okay, so you've chosen your software (eCAD, for our purposes here), worked through the tutorials/best practices, and laid out your circuit board. Now, you're ready to get them made. The first thing you have to do is export the Gerber and NC Drill files. Select the Gerber export option from the menu (**Figure 2**), open it (**Figure 3**), and then click on the Apertures button (**Figure 4**). In the Aperture window, click the Auto button to automatically name apertures (holes), then close this window. You might want to double-check your work by selecting a layer and then clicking the Preview button.

Figure 5 shows the "top" layer which is specifically the top copper layer. Notice the pads and all of the traces. The "top paste" layer (**Figure 6**) will be used for creating a stencil that we'll utilize for applying the solder paste in much the same way silk-screening is done. Finally, select Export All and save the files (without renaming them) into a folder. After exporting the Gerber files, you still need to export the NC Drill file that will tell the board house where to drill a hole and how big to make it. Just select and open the NC Drill window, click auto for naming the drill holes (**Figure 7**), and export it into the same folder that you saved the Gerbers. Now, you're ready to send your files to the board house of your choice.

Ordering Your Boards

There are a number of different board houses that you can choose from and each has a variety of options. Where you live, the size of board you're making, the number you want made, your timeline, tolerances, and how much you're willing to spend all go into deciding which vendor makes the most sense to you.

I live in the US, normally need less than three boards (prototyping), and my boards are typically less than 5 x 5

in. I'm very cost-sensitive, so I'm willing to wait two weeks or more to get my boards back. You'll need to decide what you can tolerate as far as waits and prices. Maybe have a couple different go-to places for when you might need something right away or for when you're in no hurry. As always, do searches and compare.

Once you have chosen the board house you want to use, just follow their onsite directions and upload the folder containing the Gerber and NC Drill files. Make sure you read their file submission and design rules. For example, some might require that your minimum spacing is 6 mil. That's actually fairly small. To give some perspective, the spacing between the pins on a Q44 chip (like the Propeller) is 16 mil.

Figure 6. Top paste layer preview.

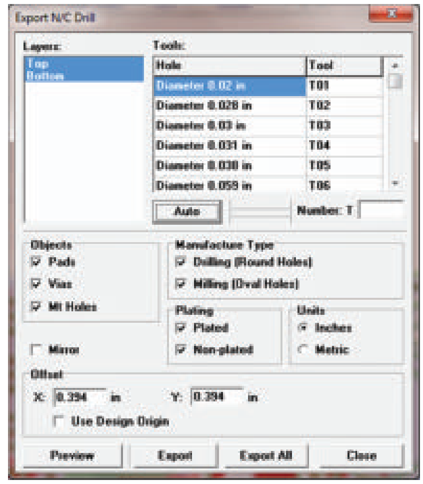


Figure 7. NC Drill Export menu.

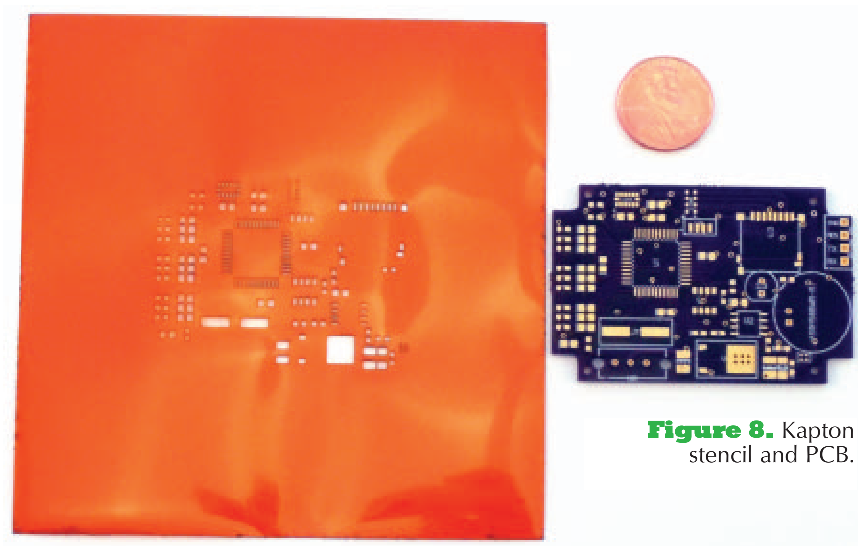


Figure 8. Kapton stencil and PCB.

Ordering Your Stencil

Ordering a solder stencil is very easy, especially if you ordered your boards from a place where the stencil is included with your order. Then, you're done already! However, if you still need to order a stencil, you have lots of places to choose from.

The first thing to decide is what material you want your stencil to be made out of: kapton or metal. Metal costs more than kapton, but will last for thousands of uses. Kapton stencils will last for hundreds of applications with proper care, and are perfect for prototyping due to their cost. A typical metal stencil will cost more than $125 plus shipping, while a kapton stencil will cost approximately $25 plus shipping. (Refer to **Figure 8**).

One nice thing you can do with 8.5" x 11" Kapton sheets (if the board house allows it) is to fit two smaller stencils on one sheet to save money.

Ordering a stencil should be very straightforward. Just follow the instructions for uploading the Gerber file which is typically called "top paste" for single-sided boards. If you're doing a two-sided board, don't forget to also upload the "bottom paste."

Ordering Solder Paste and Storage

There are a number of different varieties of solder paste; some with lead and some that are lead-free. I normally try and order my solder paste from the same place I'm getting my stencil from so that I save on shipping. Just make sure you follow all of the safety instructions and local laws if you are using lead-based solder paste. I've only used lead-based solder paste myself, so I can't comment personally as to which is easier, but I've been told that lead-free solder tends to be a bit more difficult to work with.

A number of solder paste manufacturers recommend that you store it in a cool, dry environment. This helps prolong shelf life by preventing the solder flux from evaporating. I picked up a cheap dorm room style fridge that I store my solder paste in. It is a "non-food" fridge with a sign on it marking it as such, along with the material safety datasheet (MSDS).

Holding Your PCB with a Stencil Frame

To apply the solder paste, you first need to securely

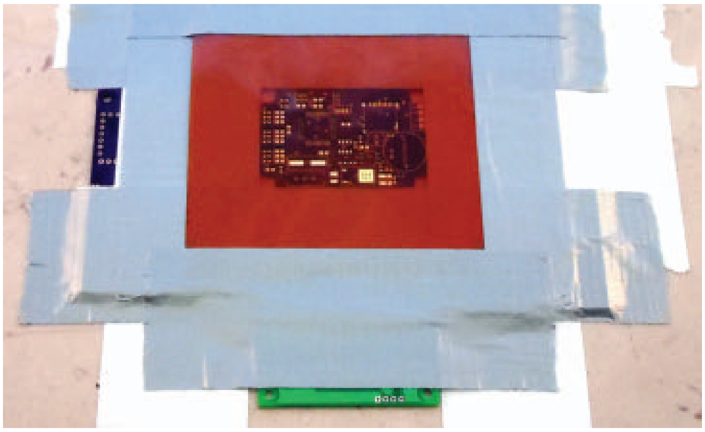


Figure 9. PCB with stencil. Notice that the PCB holder is made from old PCBs and duct taped to a flooring tile.

hold your PCB using a stencil frame. You could buy a more professional style model, or use duct tape and a few old pieces of plastic as I do. Sometimes I use a few old blank PCBs from previous projects. The important thing is that the scraps are the same thickness as your current PCB.

A friend of mine mentioned that he would tape his stencil frames to 14″ x 14″ polished flooring tiles that he had left over after remodeling. They ended up working very well for him because they provided a heavy flat surface that he could also easily store with the stencil still attached.

I followed his advice and purchased a couple of light colored polished floor tiles from a home improvement store for a few dollars. The light color lets me know where any solder paste is, and the smooth surface makes it easy to clean.

Once you have a sturdy work surface, tape your frame around the PCB. Then, line up the stencil with the solder pads on the PCB, and tape it down on one side (**Figure 9**). You're now ready to apply the solder paste.

Applying the Solder Paste

To apply solder paste, begin by cleaning the PCB and preparing the paste. I personally use 90% isopropanol and a coarse plastic brush. To prepare the solder paste, I let it warm up to room temperature and then mix it in its container until it reaches a smooth texture with a slight shine (while wearing rubber gloves).

It's important that the paste isn't dried out. If it's a bit crumbly, you can try adding some solder flux to get it back into shape. Spend some time and make sure your solder paste is nice and smooth.

Next, take a large flat edge without any dings — like a flexible paint scraper — and apply a bead of solder paste to the edge (**Figure 10**). Then, brush the paste over the stencil in one smooth motion. Carefully lift the stencil up and take the board out. Examine the pads and make sure they all have solder (**Figure 11**).

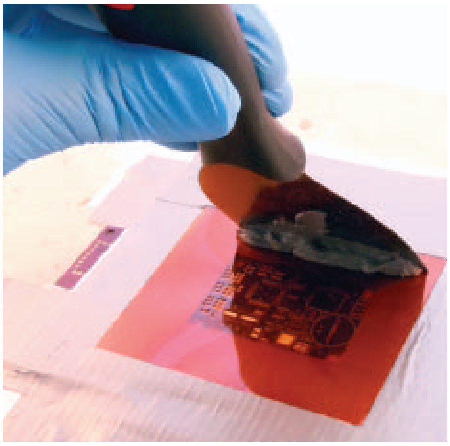


Figure 10. Paint scraper with bead of solder paste.

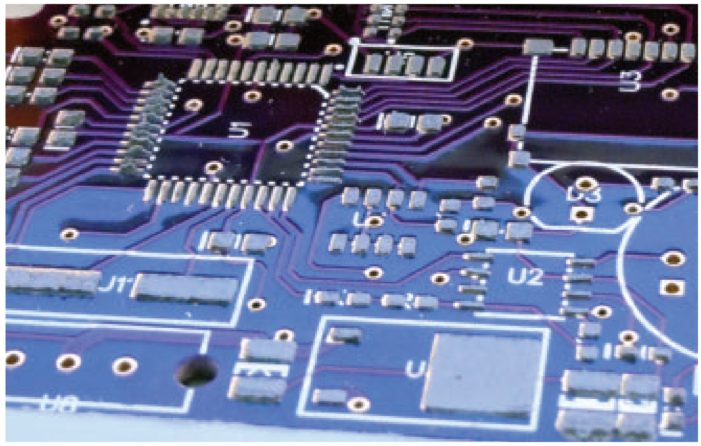


Figure 11. PCB with solder paste (gray) on pads.

Don't worry if it doesn't work the first time; just clean the board off and try again. You'll soon get the hang of it.

Placing Components

I always enjoy placing the SM components on the board. Before you begin, double-check the documents and

Figure 12. Toaster oven for reflowing SM boards.

make sure the SM components don't need any kind of special pretreatment and can tolerate the temperature of your toaster oven. Most people tend to start with the smaller components and then work their way up in size. The important thing is that you take appropriate precautions against electrostatic discharge by grounding yourself. I also like to gently press each component into the solder paste so that it doesn't pop up while the solder melts during reflow in the toaster oven. Don't worry about any of the TH components; you'll solder them in after the board is "baked."

Baking Your Board (Reflow)

There are many different ways that you can melt (reflow) the solder paste. I use an inexpensive toaster oven (**Figure 12**). Like the fridge that I store my solder paste in, the toaster oven is used *only* for reflowing PCBs boards and *never* for food. Again, be safe and only reflow in a well-ventilated area and avoid breathing in any fumes.

When I "bake" or reflow a board, I simply place it on the rack, turn the temperature up to continuous, watch for the solder paste to melt, and then turn the oven off. It's easy to tell once the solder paste has melted because it will change from dull gray to shiny silver and shimmer (**Figure 13**). Just make sure that all of the solder paste has melted — especially around the larger components that take longer to heat up.

The entire process should only take a few minutes (mine typically take 1-2 minutes when the oven is hot). It's important not to over-bake your board because — just like anything — you can burn it out.

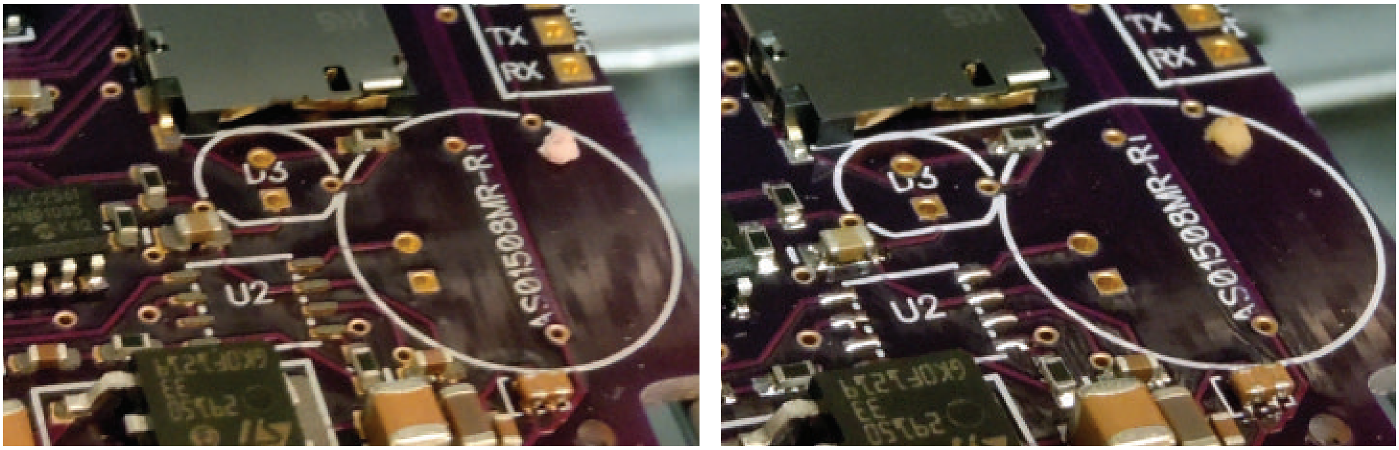


Figure 13. PCB in toaster oven, before and after. Notice how the solder changes from dull gray to shiny silver.

Conclusion

Once you've made your own circuit board, you'll never look at another one the same way again. Checking out professionally done PCBs is a great way to learn techniques and tricks.

Remember, don't be discouraged if your circuit board doesn't work the first time. Mine didn't. Just keep practicing. In fact, sharp-eyed readers will notice that there is a chip missing from the board in this article. It turned out that I didn't have that part in stock, so I had to hand-solder it in later. Errors happen. Just learn from your mistakes, and keep working on it. **NV**

By Garry Paul Spencer

Game Programming for the Arduino

Game programming for the Arduino? You're kidding, right? Nope, not a bit. Not only is it possible to write game programs for the Arduino, it is downright fun to do so.

At first, I thought my chances of being able to write a game for the Arduino would be about the same as my chances of being kicked by a unicorn in my living room on Groundhog Day. All of that changed, however, when I saw references to Seeed Studio's Touch Shield V2.0 — a 240x320 backlit touchscreen color LCD display specifically designed for the Arduino Uno microcontroller board. This got me thinking that maybe — just maybe — I might be able to write an Arduino-based game. Sure enough, shortly after unboxing my Arduino and touchscreen LCD display, I had finished writing a simple 4x4 sliding puzzle game.

Over the years, I have written code for a globzillion different microprocessors and microcontrollers — give or take a couple. I found that writing code for the Arduino was one of the easiest and most pleasant programming experiences I have ever had, thanks to their well-designed IDE (Integrated Development Environment) and hardware configuration.

According to the official Arduino website, "*Arduino code is effectively C/C++ and compiled with avr-gcc — meaning you can use all the power (and complexity) of C and some features of C++ in your sketches.*"

If you have any experience with C, C++, or any of the other "curly-brace" languages, you should have no trouble with programming the Arduino. Even novices can learn Arduino programming very quickly.

Okay, let's get started. First, you will need to purchase the microcontroller board and a touchscreen LCD display. As stated, for the sake of this article, I'm using the Seeed Studio Touch Shield version 2.0 display and the Arduino Uno Revision 3 microcontroller board. Use the links in **Table 1** to navigate directly to the appropriate product pages for these specific items.

Table 1. Product Links
www.eio.com/p-42030-arduino-uno-r3.aspx
www.eio.com/p-46564-seeedstudio-sld10261p-28-tft-touch-shield-v20.aspx

Table 2. Touch Shield Libraries
https://github.com/Seeed-Studio/TFT_Touch_Shield_V2
https://github.com/Seeed-Studio/Touch_Screen_Driver

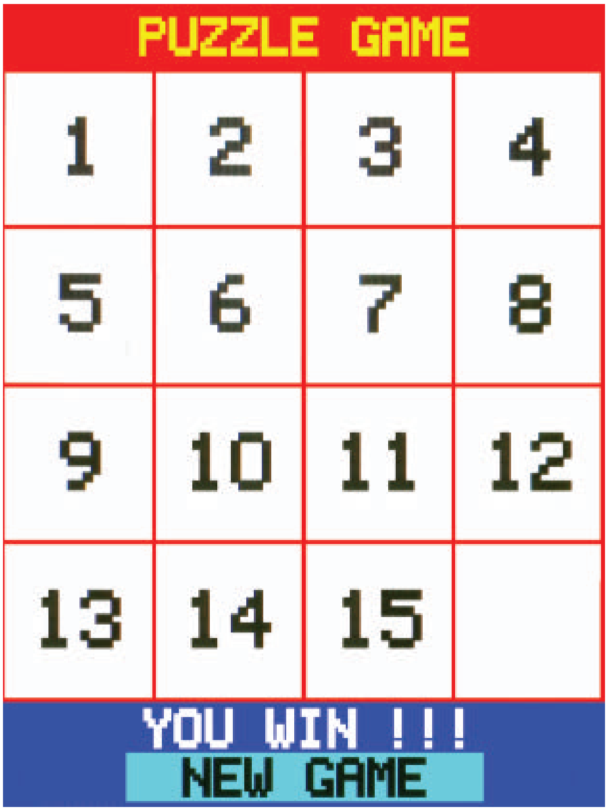


FIGURE 1.

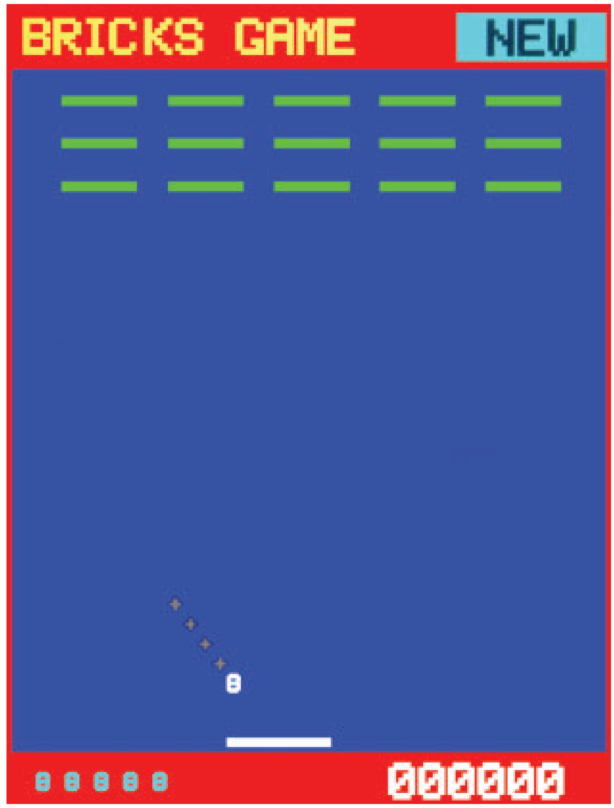


FIGURE 2.

Next, you will need to download the Arduino IDE. To do this, visit the official Arduino website (**www.arduino.cc**) and click the Download link at the top of the page. (The following instructions are for Windows users.) Click on the Windows Installer link and run the installation program. This will cause the Arduino IDE to be installed in the C:\Program Files (x86)\Arduino folder.

Now, download *both* of the Touch Shield V2.0 libraries. Visit each link in **Table 2** with your web browser, and click the Download ZIP button on each page. Open each of the ZIP archives and unzip all of the contents into the C:\Program Files (x86)\Arduino\libraries folder. Because of a quirk in the way the Arduino IDE imports libraries, you will need to rename both of the folders you just created. Rename the TFT_Touch_Shield_V2-master folder to TFTscreen; rename the Touch_Screen_Driver-master folder to TFTtouch.

As you can see from the screenshots in **Figures 1** and **2**, the touchscreen LCD display was designed to be used in portrait mode (240x320) instead of landscape mode. As near as I can tell, there doesn't seem to be any way to change the orientation, so you need to take this into consideration when designing games for this display. I mainly used font sizes 2, 3, and 4. Font size 1 is pretty much useless because the characters are so small they are virtually unreadable. Although finger pressure detection is very good on the touchscreen, I would recommend that you use a stylus because it is more precise and it obscures the screen less than a person's finger. This can be a factor in playing some of the games.

Let's look at the code for the sliding puzzle game which is shown in **Table 3**. You will notice that the variable declarations appear at the top of the program. There are two main sections in Arduino code: the "setup" block and the "loop" block. The setup block is meant to hold code that is executed only once. The code in the loop block is executed over and over again.

After the loop block, I put the custom functions that were used to handle specific tasks. In the "setup" block, I wrote some code that draws the playing grid for the puzzle game using the library functions provided by Seeed Studio.

In your program code, each screen function name should be preceded by the "Tft" library object name. For example, *Tft.fillScreen(0,239,0,319,BLUE);* would fill the entire screen with the color blue. Here are some more examples:

Tft.drawLine(unsigned int x0,unsigned int y0,unsigned int x1,unsigned int y1,unsigned int color);

Draws a line from the upper-left point (x0,y0) to the lower-right point (x1,y1) using the specified color.

Tft.drawRectangle(INT16U poX,

Table 3. Available at **www.ArduinoGames.com/puzzle.zip**.

```
#include <stdint.h>
#include <SeeedTouchScreen.h>
#include <TFTv2.h>
#include <SPI.h>
#define YP A2
#define XM A1
#define YM 14
#define XP 17

TouchScreen ts = TouchScreen(XP, YP, XM, YM);
int bc; int br; int c; int c1; int c2; int r;
int r1; int r2; int rsd=0; int tc; int tr; int
x; int y;
int b[5][5]; String s[]={"
","1","2","3","4","5","6","7","8","9","10","11",
"12","13","14","15","  "};

void setup() {
Tft.TFTinit(); TFT_BL_ON;
 // Init TFT library and turn on the backlight
Tft.fillScreen(0,239,0,319,BLACK);
Tft.drawString("SLIDING PUZZLE
GAME",4,10,2,YELLOW);
Tft.drawString("TOUCH THE
SCREEN",20,160,2,CYAN); Tft.drawString("TO BEGIN
PLAYING",20,180,2,CYAN);
touchWait(); randomSeed(rsd);
Tft.fillScreen(0,238,0,319,BLUE);
Tft.fillScreen(1,237,1,25,RED);
Tft.drawString("SLIDING PUZZLE
GAME",4,6,2,YELLOW);
Tft.fillScreen(1,237,26,262,WHITE);
Tft.drawLine(1,26,237,26,RED);
Tft.drawLine(1,85,237,85,RED);
Tft.drawLine(1,144,237,144,RED);
Tft.drawLine(1,203,237,203,RED);
Tft.drawLine(1,262,237,262,RED);
Tft.drawLine(1,26,1,262,RED);
Tft.drawLine(60,26,60,262,RED);
Tft.drawLine(119,26,119,262,RED);
Tft.drawLine(178,26,178,262,RED);
Tft.drawLine(237,26,237,262,RED);
Tft.fillScreen(50,189,297,317,CYAN);
Tft.drawString("NEW GAME",70,300,2,BLACK);
newGame();
}

void loop(void) {
touchWait(); if (touchBox(50,297,189,317))
{newGame();}
if (touchBox(1,26,236,261))
 {findBlank();
  for (r=1;r<5;++r) {for (c=1;c<5;++c) {int
  yy=(59*r)-33; int xx=(59*c)-58;
  if (touchBox(xx,yy,(xx+58),(yy+58))) {tr=r;
  tc=c;}}}
  if (((tr == br) && (tc != bc)) || ((tr != br)
  && (tc == bc))) {cellMove();
  for (r=1;r<5;++r) {for (c=1;c<5;++c)
  {cellShow(r,c);}}}
  if (inOrder()) {Tft.drawString("YOU WIN
  !!!",55,272,2,WHITE); touchWait(); newGame();}
 }
}

void cellMove(void) {
while (tr<br) {b[br][bc]=b[br-1][bc]; b
[br-1][bc]=16; --br;}
while (tr>br) {b[br][bc]=b[br+1][bc];
b[br+1][bc]=16; ++br;}
while (tc<bc) {b[br][bc]=b[br][bc-1]; b[br]
[bc-1]=16; --bc;}
while (tc>bc) {b[br][bc]=b[br][bc+1];
b[br][bc+1]=16; ++bc;}
}

void cellShow(int rrr, int ccc) {
String ss=s[b[rrr][ccc]]; int ssL=ss.length()-1;
int xx=(59*c)-57;
int yy=(59*r)-32;
Tft.fillScreen(xx,xx+55,yy,yy+55,WHITE);
char sss[]={0,0,0}; for (int i=0;i<=ssL;++i)
{sss[i]=ss.charAt(i);}
Tft.drawString(sss,xx+19-(ssL*10),yy+17,
3,BLACK);
}

void findBlank(void) {
for (r=1;r<5;++r) {for (c=1;c<5;++c) {if
(b[r][c] == 16) {br=r; bc=c;}}}}

boolean inOrder(void) {
boolean ordered=true;
for (r=1;r<5;++r) {for (c=1;c<5;++c) {if
((b[r][c]) != (((r-1)*4)+c)) {ordered=false;}}}
return ordered;
}

void newGame(void) {
Tft.drawString("YOU WIN !!!",55,272,2,BLUE);
for (r=1;r<5;++r) {for (c=1;c<5;++c)
{b[r][c]=((r-1)*4)+c;}}
int rc=0; findBlank();
for (int i=1;i<100;++i)
 {rc=1-rc; if (rc == 0) {tr=br; tc=random(1,5);}
 else {tr=random(1,5); tc=bc;}
  cellMove();
 }
for (r=1;r<5;++r) {for (c=1;c<5;++c)
{cellShow(r,c);}} findBlank();
}

boolean touchBox(int x0, int y0, int x1, int y1)
{
if (x<x0 || x>x1 || y<y0 || y>y1) {return
false;} return true;}

boolean touchPosition() {
Point p = ts.getPoint();
if (p.z > __PRESURE)
 {x = map(p.x, TS_MINX, TS_MAXX, 0, 240);
  y = map(p.y, TS_MINY, TS_MAXY, 0, 320);
  return true;} else {return false;}
}

boolean touchStatus(void) {
int steady=0; boolean newState=false; boolean
oldState;
while (steady <20)
  {oldState=newState; Point p = ts.getPoint();
  if (p.z > __PRESURE)
    {newState=true; x = map(p.x, TS_MINX,
    TS_MAXX, 0, 240);
     y = map(p.y, TS_MINY, TS_MAXY, 0, 320);}
     else {newState=false;}
   if (newState == oldState) {++steady;} else
   {steady=0;}
   delay(5);
  }
return newState;
}

void touchWait(void) {
while (!touchStatus()) {++rsd; if (rsd>10000)
{rsd=0;}}
while (touchStatus()) {}
}
```

INT16U poY, INT16U length, INT16U width, INT16U color);

Draws a rectangle with a line width of one pixel starting at the upper-left corner (poX,poY) and having the specified length, width, and color. The *INT16U* means unsigned 16-bit integer.

*Tft.drawString(char *string, INT16U poX, INT16U poY, INT16U size, INT16U fgcolor);*

Places a character string on the screen at (poX,poY) with the specified color and font size. Note that this function can accept a literal string or a null-terminated character array, but NOT a string variable.

Tft.fillScreen(INT16U XL, INT16U XR, INT16U YU, INT16U YD, INT16U color);

Fills a portion of the screen with the specified color from the upper-left corner (XL,YU) to the lower-right corner (XR,YD).

Tft.setPixel(INT16U poX, INT16U poY, INT16U color);

Draws a single pixel at (poX,poY) in the specified color.

This display uses 16-bit colors having a binary form of RRRRRGGGGGGBBBBB, where R is red, G is green, and B is blue. For example, pure red is 1111100000000000 in binary or 0xF800 in hexadecimal. The LCD screen library defines the following (case-sensitive) colors: BLACK (0x00), WHITE (0xFFFF), BLUE (0x1F), BRIGHT_RED (0xF810), RED (0xF800), GREEN (0x7E0), CYAN (0x7FF), YELLOW (0xFFE0), GRAY1 (0x8410), and GRAY2 (0x4208).

Some degree of randomness is necessary in most games. In the Arduino system, the random number generator must be seeded. Otherwise, it will return the same sequence of numbers each time the program is executed. I decided to use the length of time the user waits before first touching the screen as the seed for the random number generator. At the start of each round of play, the random numbers are used to scramble an initially ordered playing board. This is done in the *newGame* function.

I wrote three functions for working with the touch interface of the LCD screen: *touchPosition*, *touchStatus*, and *touchWait*. The *touchStatus* function returns *true* if the user is touching the screen and *false* if not.

You will notice that the touch interface is polled until 20 consecutive identical responses are received. This will eliminate any unreliable results. If your game

opening salvo such as:

"Last time we met, we enlisted the aid of a couple of old standbys — the 74-165 and 74-595 shift registers — to expand I/O in our projects. We even managed to craft some PASM code to provide PWM on multiple x595 outputs ... "

Well, by then, my eyes were glazing over and I searched for some lexical help, but there was none to be found.

If you truly intend to capture the interest of (well, for example, me) those to whom you make reference in the magazine editorial, then please give us a fighting chance with some definitions.

One the other hand, the Fred Eady article was easily graspable, as was the article by Joe Pardue. There were more that I could grasp, but when a secret language is used, I fear you will only preach to the choir and thereby limit your readers to those who know the secret knowledge.

Don't slip on a slithey tove!

Jack Leissring

ERRATA

I have gotten several good comments about the solar design article featured on the March 2014 cover. However, one reader pointed out that the MOSFET has a reversed diode and will not totally turn off, which could allow the battery to drain into the solar panel. I totally missed this on the part I specified and ask forgiveness.

Some solar cells come with a blocking diode built in and others don't. To prevent discharge, I have added a blocking diode (not included in the kit) which should be placed in line with the positive wire from the solar panel. Use a Fairchild SB1245 or equivalent. Please see the corrected schematic at the article link on the *NV* website. The cathode should go toward the board.

Due to this change, I have also revised the code. The revision is at the link, as well.

There is an error in the published parts list regarding R5 and R6. The correct value is 910 ohms and has been corrected on the parts and source list on the website.

R1 and R3 are correct with their values at 8,200 ohms. However, the published part number is incorrect and will give you 18.2K. The correct part number is 271-8.2K-RC and has been corrected also.

NOTE: If you put a voltmeter on the battery, it will reflect the charging voltage and not the battery voltage. Always keep in mind that the higher the voltage to the battery, the more power is being put into it. Make sure you fuse the unit.

One reader noted that you should be able to parallel two MOSFETs being driven by the same circuit. I had thought about this when writing the article, however, wanted to take one step at a time.

Several readers pointed out that when the solar panel goes below the battery voltage, a buck boost circuit could be used to continue the charging. This was not called for in the specifications and would probably increase the cost 3-4 times due to the size of the components needed.

Ron Newton

MakerPlot – Part 8
The DIY Software Kit

By John Gavlik
and Martin Hebel

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/ index.php?/magazine/article/may2014_MakerPlot**.

In previous articles, we've shown you what MakerPlot can do right out of the box. Now, we're going to show you what MakerPlot can become in your hands because we're going to give you all the tools you'll need to construct whatever GUI you want in order to capture, plot, log, and manipulate your micro's analog and digital data.

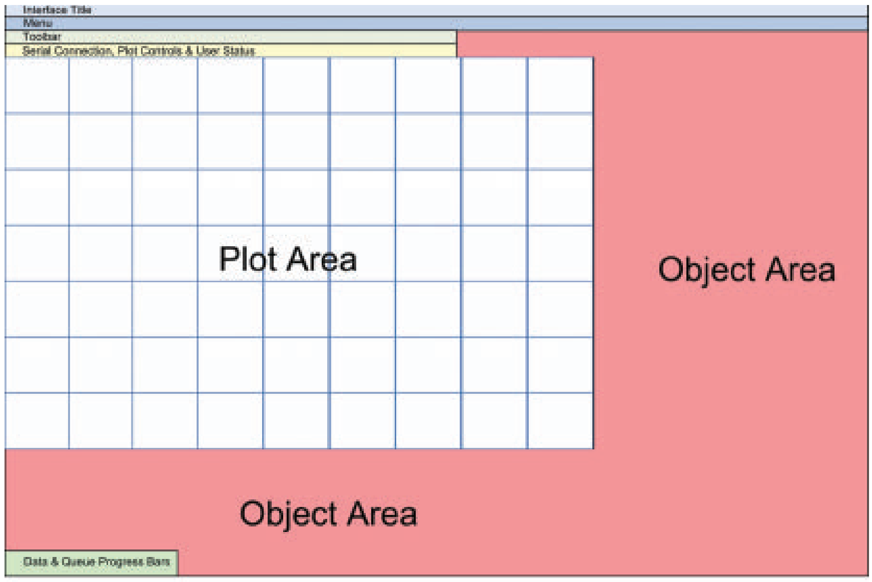


Figure 1. MakerPlot interface areas.

This is the first in a series of articles on how to customize MakerPlot. By its very name, MakerPlot hints at this customization feature and, of course, that's why we call it "The DIY Software Kit." Unlike most hardware kits that have only one or maybe two primary functions, MakerPlot is an open-ended software GUI that has no practical limits in terms of what can be built.

As always, if you haven't already done so, you can download a free 30 day trial copy of MakerPlot from **www.makerplot.com** to follow along. If you like what you see and what it does, ***you can order it from the NV Webstore at a discounted price***. Let's get going.

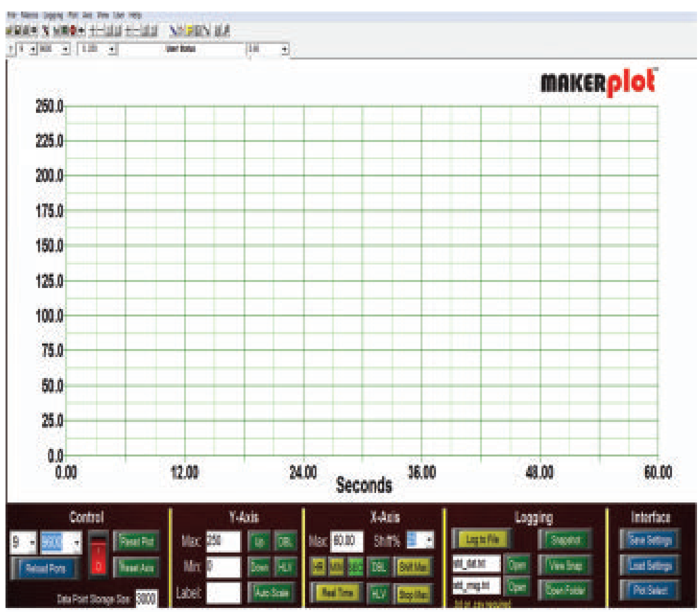


Figure 2. The standard interface.

Interface Basics

If you're new to these articles, we refer to the GUI as an interface that's populated with controls like meters, buttons, switches, text boxes, and one or more plot areas. **Figure 1** is an example of the discrete screen areas within the MakerPlot GUI. Line and bar graph plotting occurs in the plot area(s), while the object area(s) are where the

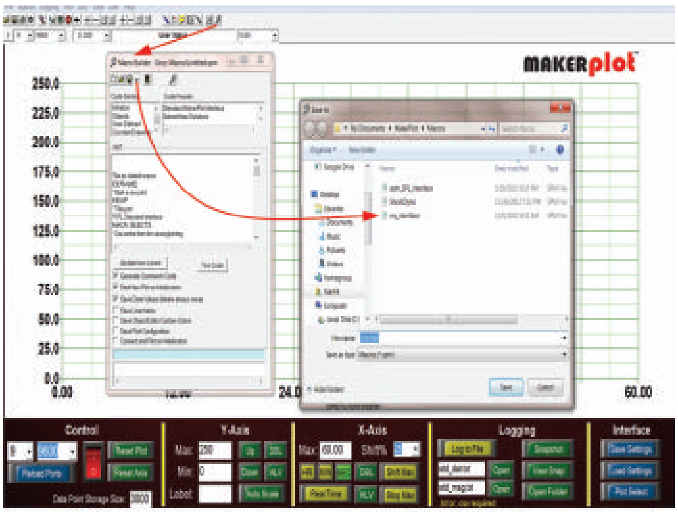


Figure 3. The Macro Builder.

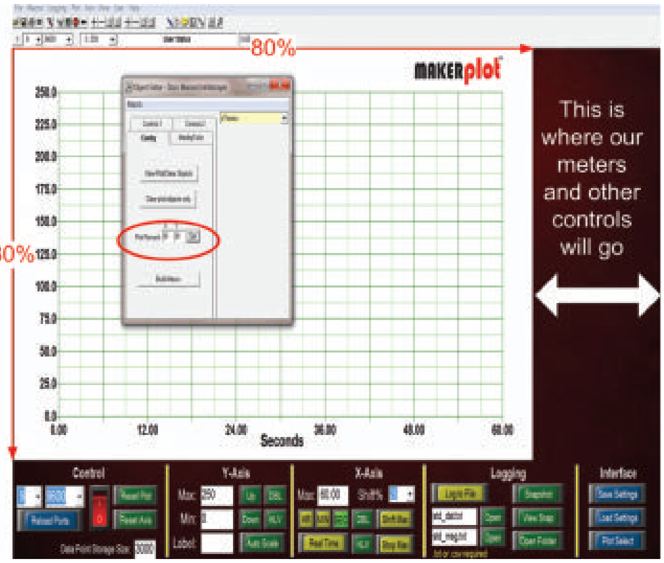


Figure 4. Setting plot area dimensions.

controls are placed. In MakerPlot speak, controls are things like meters, buttons, switches, labels, sliders, text boxes, etc. At the very top are the drop-down menus, toolbar icons, and the connection and user status areas. At the bottom are the real time data and queue progress bars that monitor the incoming data from your micro and from other sources like mouse clicks on the interface. So, that's the general composition of the MakerPlot interface. However, for customization purposes, only the plot and object areas can be manipulated.

Creating a New Interface

Rather than start from scratch to develop a custom interface, let's start with our standard interface **(Figure 2)** as it has a complete set of menu buttons and text boxes on the bottom, and a wide plot area. What it doesn't have are meters, sliders, and other controls that are usually on the right side. To make room for these other controls, we'll need to adjust the plot area to expose more object area. Here's how it's done.

Referring to **Figure 3**, we need to save the standard interface under a different file name to keep it separate from our customization efforts. To do this, click on the Macro Builder icon (it's the one that looks like a wrench) and the Macro Builder drop-down menu will appear. Click on the Save As down arrow and save it under the new file name *My_Interface.spm*. Now, we can begin to experiment without affecting the standard interface**.**

In order to make room for other controls, we'll need to reduce the size of the plot area. To do this, click on the Object Editor icon (it's the one that looks like a hammer) and its drop-down menu will appear. In the Config tab, set

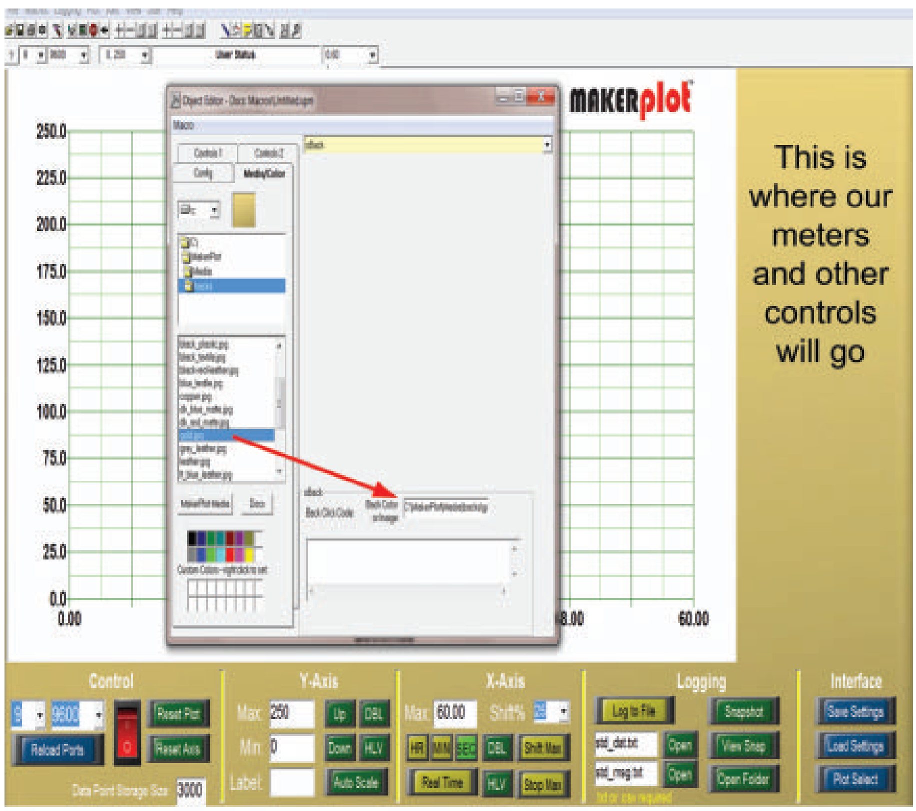


Figure 5. Changing the background color.

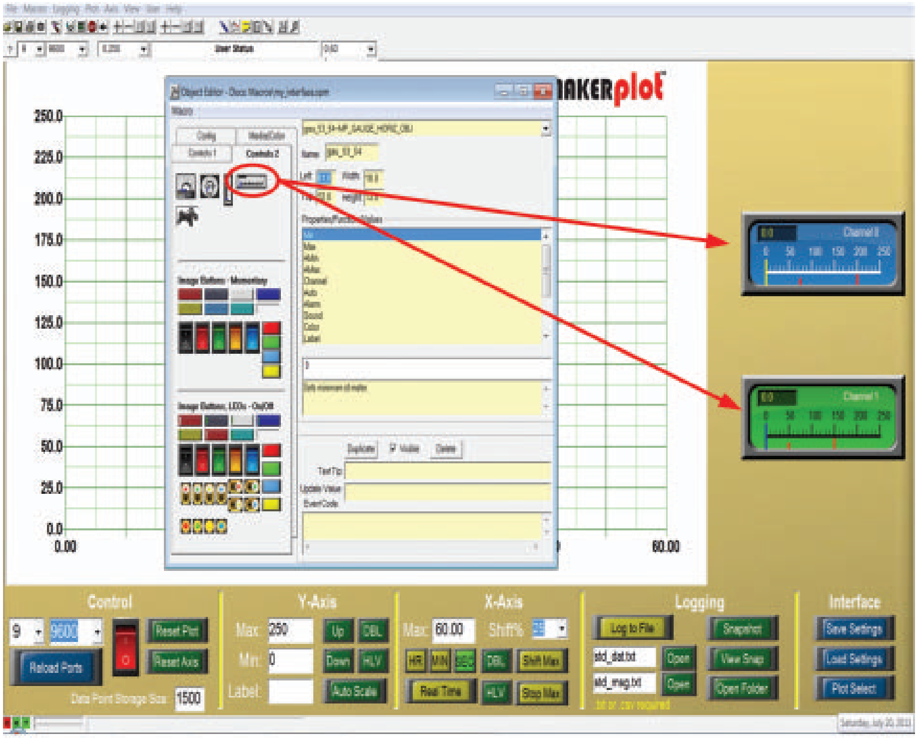


Figure 6. Adding two horizontal meters.

the plot X and Y percentages to 80 and 80, which means 80% in both the X and Y directions for the plot area (**Figure 4**). Now, there's room on the right to place meters and other controls. Before we do this, let's see how to change the background color from a dark red maroon to gold (**Figure 5**). Using the Object Editor, click on the Media/Color tab and select oBack, which stands for Object Background. Next, click on the gold.jpg from the drop-down list and drag it to the rectangular area as shown. This will change the color of the Object Area background to gold. The choices of colors and background jpgs run the gamut, so you can select whatever suits your fancy.

Adding Meters

So far, so good. Now, let's add two horizontal meters.

Referring to **Figure 6,** click on the Object Editor toolbar icon and click on the Controls 2 tab. At the top, you'll see four types of meters: square, round, vertical, and horizontal. Left-click and hold the mouse cursor on the horizontal meter and drag it to the top-right of the Object Area and release. This will coarsely position the meter.

Next, click on the Duplicate button in the Object Editor as this will bring up another instance of a horizontal meter.

You can reposition both meters separately using the keyboard arrow keys. You can also make changes to the meter's background colors by using the Object Editor for that meter; we've chosen blue for the top meter and green for the bottom one to distinguish them from one another.

Finally, you can add minimum and maximum alarm settings (the little red marks under the meter scale), as well. When the analog signals that the meters are linked to go above or below these settings, an audio alarm will sound. Plus, you can change the type of audio alarm to know which meter is responding.

We're purposely leaving out a lot of the placement, positioning, and other details to avoid boring you, but if you want to know exactly how all of this is done, then view the videos at **www.makerplot.com** under **Maker Videos ➔ Adding Meters**.

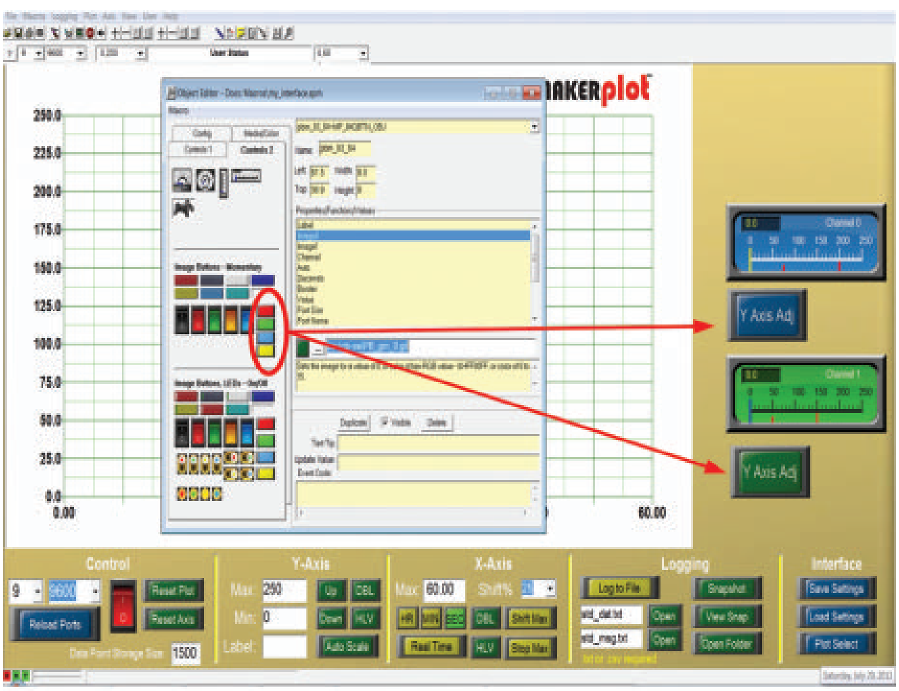


Figure 7. Adding Y axis adj buttons.

Adding Buttons and Switches

The next thing to do is to add some buttons and switches to control our meters.

For example, the meter scale now reads 0 to 250, but maybe our plot area's Y scale is set at 0 to 1,000. Let's add a momentary pushbutton below each meter to adjust the meter scale to the Y axis scale. This is done by selecting one of the Momentary Image Button icons and dragging it under the meter. Then, label it "Y Axis Adj." When you click on this button, it will set the meter scale to the Y axis (**Figure 7**).

Recall that both of these meters have alarm settings and if the alarms are exceeded on either the high or low end, the meters will "beep" or sound whatever audio alarm you select for each of them. To mute the audio tones, we'll add a slider switch to each meter so when the switch is ON, the tone will sound. When it's OFF, it won't. It's basically the same procedure: Left-click on the slider switch and drag it under the meter, then position it with the keyboard arrow keys like the Y axis switches (**Figure 8**).

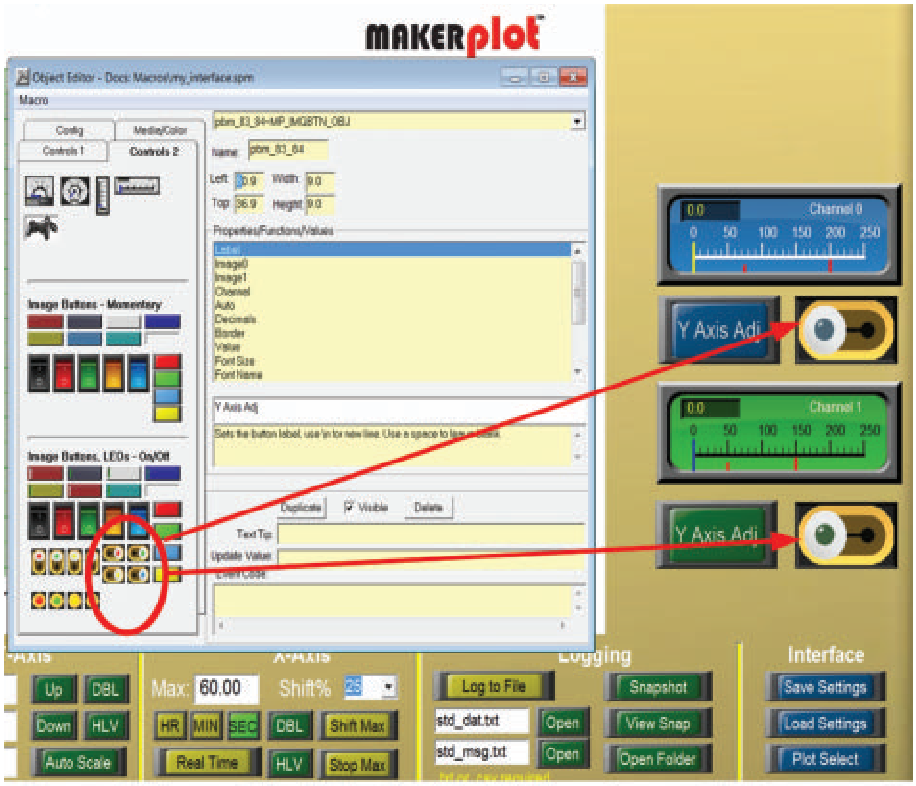


Figure 8. Adding meter alarm switches.

We've adjusted each meter to monitor a single analog channel, with the top meter set to analog Channel 1 and the bottom meter set to analog Channel 0. Now, when we begin plotting analog data again, these meters will display the levels of the analog signals.

Well, so much for analog signals. How about digital?

LEDs and More

For monitoring individual digital signals, LEDs are usually the choice. While MakerPlot is capable of displaying up to 32 separate digital signals, this example is limited to eight LEDs. Using the Object Editor again, we've clicked on and dragged a single red LED and placed it above the top meter. Then, we used the Duplicate button to create seven more.

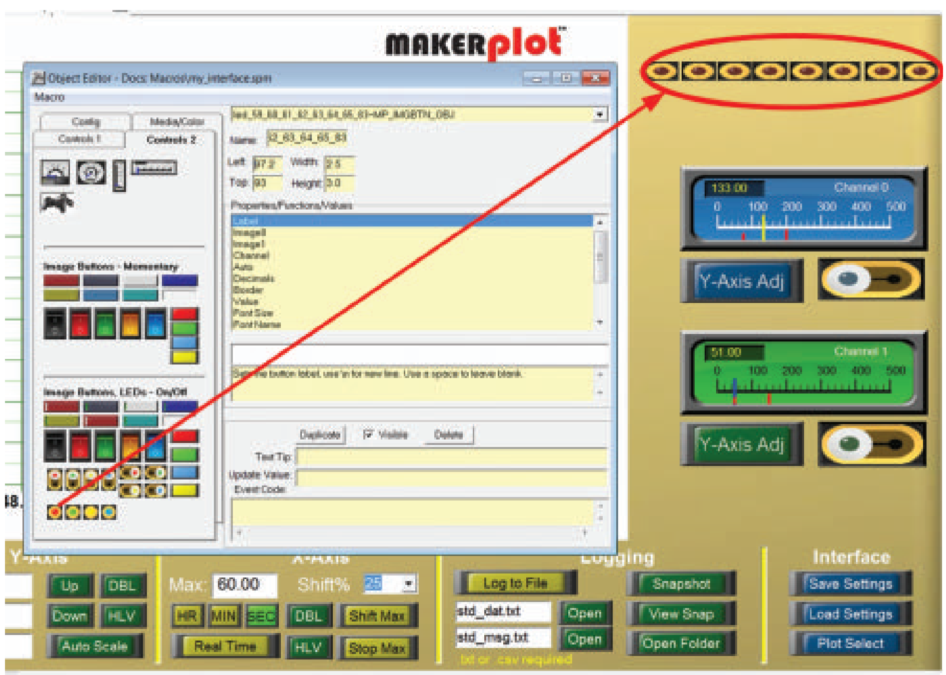


Figure 9. Adding LEDs.

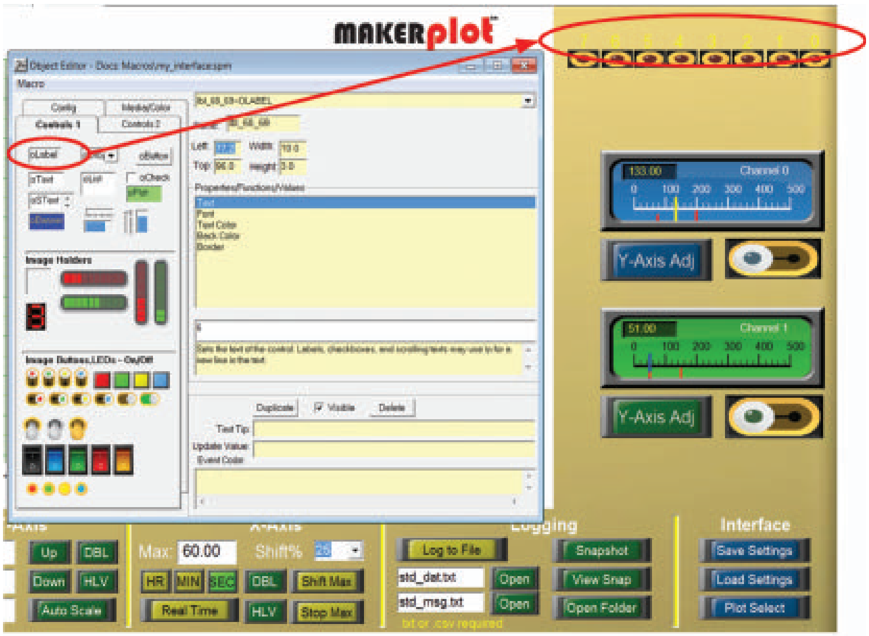


Figure 10. Adding LED labels.

Finally, we arranged all of them in a row using the keyboard arrow keys; the result looks like **Figure 9**.

Next, we labeled each LED with its corresponding binary weight from 7 (MSB) on the left to 0 (LSB) on the far right. This time, we switched to the Controls 1 tab to select the oLabel; using the Text Box in the drop-down menu, we set the numbers from 7 to 0 as in **Figure 10**. That configures the individual LEDs, but we also want to see the numerical value of the eight digital signals, so that requires a Text Box. To do this, we grabbed an oText box and dragged it under the LEDs to display the numerical value (**Figure 11**). Then, we dragged another oLabel and named it Binary Value.

So, that's about it for the digital signals. Now, let's add some window dressing to our customized Interface.

Dressing Things Up

While the meters and buttons for analog and digital signal monitoring are now on the *My_Interface.spm*, what remains is a way to finish things off. Let's do that with some yellow vertical and horizontal borders. In our last example (**Figure 11**), we used an oLabel control as in Binary Value to label the oText box that displays the real time numerical value of the digital signals. As it turns out, we can use a similar oLabel control for horizontal and vertical boards, too; it just needs to be resized. As a matter of fact, the vertical yellow bars that separate the menu controls on the bottom are all really oLabel controls, so we're going to use them as examples for vertical borders.

To do this, simply do a shift-right-click on any of the vertical bars and click the Duplicate button. This will bring up more of the same. Next, use the arrow keys to resize them and move them into place. The result should look like **Figure 12** when complete. You'll notice that the meters, buttons, and switches are now bordered by the same horizontal and vertical bars as the menu area.

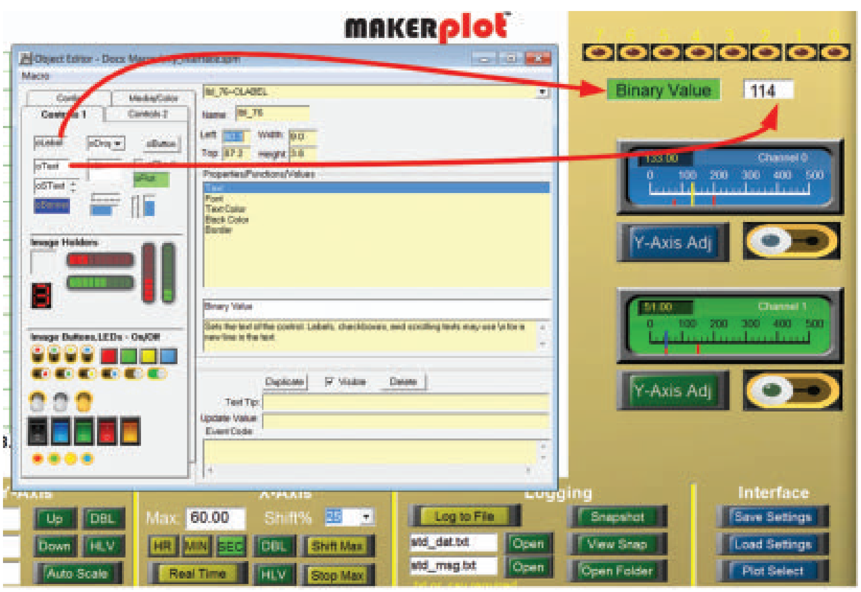


Figure 11. Adding Binary Value.

Saving Our Work

While the graphic part of the interface is complete, we're not done yet. What remains is to save our work to the *My_Interface.spm* file. To start, you'll need to click on the Macro Builder icon in the tool bar to bring up the Macro Builder drop-down menu (**Figure 13**). Then, click on the Update from Current button that will load in all the graphic code for the meters, buttons, switches, labels, and text boxes that have been added. What's left is to click on the traditional floppy disk icon to save our work to the *My_Interface.spm* file.

Conclusion

That about does it for building the graphics portion of our *My_Interface.spm*

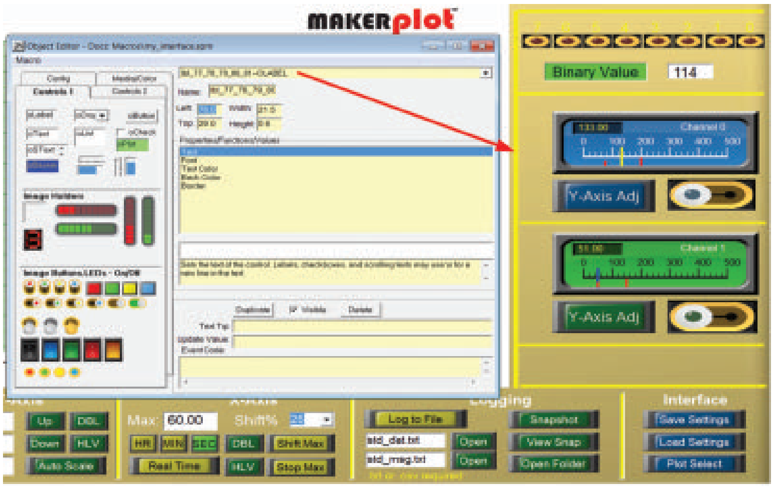


Figure 12. Adding borders.

file. What still remains is adding MakerPlot instructions to the meters, switches, and text boxes in order to make these graphics do what we want them to do. We call these Event Codes. We'll get into this next time because it's just as important as the graphics we just showed you.

So, in review, you've been shown how to build a custom interface using the Object Editor and Macro Builder — two of the main building blocks of MakerPlot. Because of the details involved, we're not going to go much further into how these two menus work in these articles, so, we invite you to find out more about them in our Video Tutorial Series. Just go to **Maker Videos ➔ Object Editor Part 1 and 2** and **➔ Macro Builder**.

Of course, all the details of everything about MakerPlot can be found in our MakerPlot Guide. By the end of this series, you will at least be aware of all the things you can accomplish yourself to customize MakerPlot; then it's up to you to roll your own version. That's all for now, so just remember: *Got Data – MakerPlot It!* **NV**

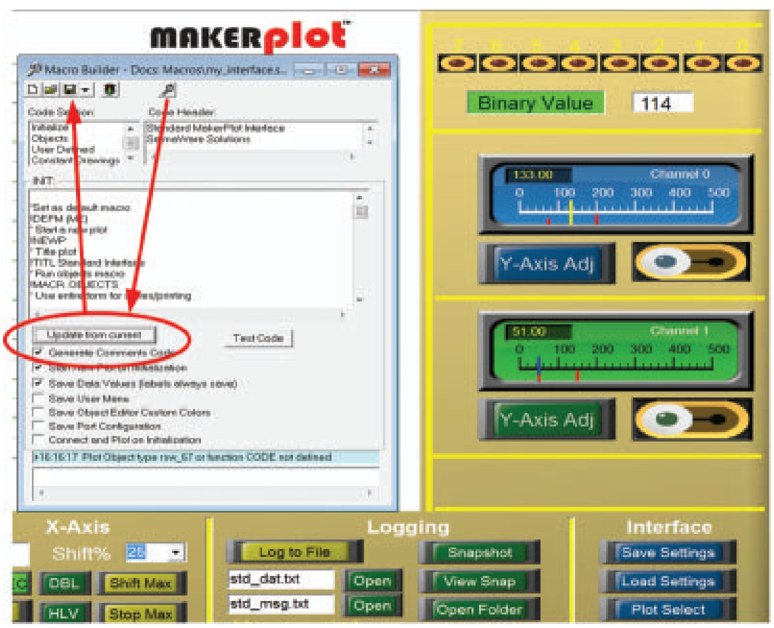


Figure 13. Saving our work.

Oh, Say, Can You I²C?

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2014_SpinZone**.

At the risk of stating the blindingly obvious, we human beings are an interesting lot. Our ability to live in denial and make silly assumptions is remarkable. My acting teacher — the late Cliff Osmond — used to point out that we all know we're going to die, yet we live our lives as though that day will never come. As electronics and programming enthusiasts, we often assume that if something is really simple to us, then, of course, everybody else knows and understands it too. Yet, this is not always the case. In fact, it's frequently *not* the case at all. If I'm honest, I'm a little guilty in my Propeller forums posts of treating some topics as if everyone knows and understands them. I²C has been one of those topics.

I've been a fan of I²C since my days with the BASIC Stamp (the BS2p family introduced I²C instructions to PBASIC). The I²C protocol is such an integral part of the Propeller I kind of assumed everybody understands and is as comfortable with it as I am. Then, an odd thing happened. I got a ton of private messages through the Parallax forums by other enthusiasts that were struggling with I²C.

Let's see if we can do something about that, shall we?

Propeller Boot-Up Sequence

We're going to start with the boot-up sequence because an I²C EEPROM (24LC256 or larger) is connected to the Propeller to store the program and non-volatile data. Pins 28 and 29 are used for the I²C buss that connects to the EEPROM. Pin 28 is SCL (clock); pin 29 is SDA (data). Once you're comfortable with I²C, you can connect other devices to these pins.

When the Propeller comes out of a reset, it checks to see if that reset was caused by an IDE (Integrated Development Environment) wanting to download new code. This is a serial transaction on pins 30 (TX) and 31 (RX). If no IDE is detected, the Propeller will copy 32K of data from the EEPROM into the hub RAM. The boot loader copies a Spin interpreter into cog 0, and the program takes off from there.

This sequence after reset is the only time that the Propeller demands control of pins 28/29 (I²C) and 30/31 (serial). That said, I don't think it's a good idea to assign these pins to activities other than what they do in the boot sequence. That is to say that I always use pins 28/29 for I²C if I need it, and pins 30/31 for serial (usually debugging) if I need it.

The serial pins are a point-to-point connection, but I²C is a multi-drop buss. We can take full advantage of this.

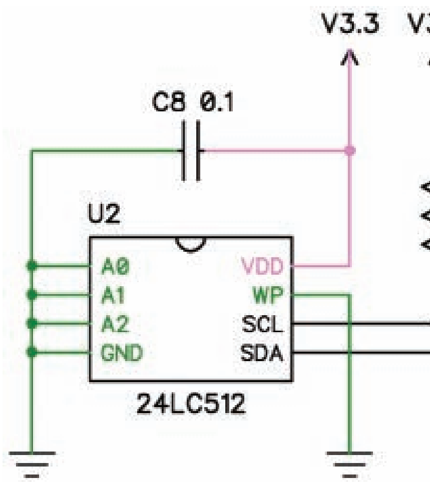
I²C Basics

First things first (*and another blindingly obvious statement considering the previous section*). I²C is a two-wire synchronous communications protocol. What this means is that the SCL line controls the movement of data on the buss; speed of the transaction is determined by the bit rate of the SCL line. You'll typically see parts specified at 100 kHz or 400 kHz — some even higher.

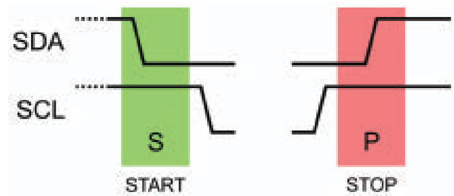
To keep things easy we're going to write an I²C driver in Spin, so it will be slower. That's okay; the synchronous nature of the buss allows for this.

The I²C protocol does support "multi-master" systems, but we're not going to get into that. In our code, the Propeller is the master; the EEPROM and anything else we hang on the buss are considered slaves. The master initiates and controls the transactions on the buss. Slaves are identified by a device type and (usually) a three-bit address. The combination of device type and address is called the control byte or slave ID.

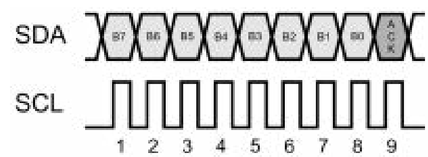
For example, the EEPROM used for holding the program is at address %000. If we wanted additional storage space, we could hang another EEPROM on the buss and set its address to %001.



■ FIGURE 1. EE connections.



■ FIGURE 2. Start and stop conditions.



■ FIGURE 3. I²C byte on SDA and SCL.

I just made the statement, " ... hang on the buss" and it should get a little explanation. The I²C physical layer defines the SCL and SDA lines as open-drain; these lines are intended to be pulled up to Vdd (typically through 4.7K to 10K). To output a one, the device floats the pin (makes it an input); this allows the pull-up to pull the line high (1). To output a zero, the device makes the pin an output and low (0).

The reason for this is electrical safety. If two devices attempt to write at the same time, the transmission will be clobbered, but with neither driving the line high while the other is actively pulling it low, there is no danger of a short circuit between devices.

Here's the rub ... the Propeller boot loader violates this specification. If you look on some older Propeller designs, you'll see that the SCL line is missing a pull-up. The reason is that the Propeller boot loader assumes the only thing connected to it is an EEPROM and it's safe to drive the clock line high and low. This isn't going to change in the Propeller because the boot loader is masked into the chip.

Don't let this worry you, though; I've built lots and lots of boards and never had an I²C device failure. Rest assured that newer Propeller designs from Parallax — like the Propeller Activity Board which is one of my favorite little development tools — have a pull-up on SCL, as well as SDA.

Please ... in your own designs, you should put the pull-up on the SCL pin because most I²C code counts on it. For reference, **Figure 1** shows the EEPROM connections from my project starter schematic (the DipTrace file is included with the downloads). If you have an older board without the SCL pull-up, there is an excellent I²C object by Mike Green that does drive the SCL line high and low.

Okay, then. Let's get into the nuts and bolts of I²C communications. An I²C transaction is built from four elements:

- Start condition
- Write byte(s)
- Read bytes(s)
- Stop condition

Yes, that's really it. Everything else is built with these functional blocks.

When the I²C master (Propeller) is instantiated, the SCL and SDA pins are set to input mode which allows the pull-ups to take both lines high. A Start condition (notated with an S in I²C transaction diagrams) is created by taking the SDA line low while the SCL line remains high (**Figure 2**).

Write and Read transactions are byte-oriented, but use nine clock cycles (**Figure 3**). The ninth clock is for the Acknowledge bit provided by the receiving end of the transaction. The master provides all clock cycles.

When writing a byte, each bit is output to the SDA line (MSBFIRST) and then the SCL line is taken high (via the pull-up), then back low. After the eighth bit, the SDA line is set to input mode and the clock is taken back high. At this point, the master samples the SDA line for ACK (0) or NAK (1).

The Read transaction is similar except the SDA line is in input mode for the first eight bits. The leading edge of the clock signal causes the slave to output a bit which is then sampled by the master. After the eighth bit, the master will output the appropriate ACK/NAK bit for the slave and pulse the clock a final time.

The Stop condition (notated with a **P** in I²C transaction diagrams) is created by allowing the SDA line to go high while the clock line is already high. Note that the Stop condition is — at times — used in the middle of a transaction. We'll see this in detail later.

Essential I²C in Spin

As I mentioned, if speed is not an issue then it's very

easy to code an essential I²C object in Spin. So, let's jump in.

Off the bat, we're going to change the name of our object's entry point. While we would traditionally use **start()**, this name conflicts with a fundamental process that is part of I²C. We'll use **setup()** as it's descriptive. Just remember that change when using the I²C object.

There are two variations of **setup()**: standard and explicit. The first uses the Propeller's I²C pins; the other uses pins of our choice. This allows us to have multiple I²C busses in a project. That doesn't happen often, but it's possible. For example, I created a little DS1307 clock module for the EFX-TEK HC-8+ controller. That controller makes four user I/O pins available, but not the I²C buss. This is not a problem as the object allows us to define the SCL and SDA pins. Note how the **setup()** method actually calls **setupx()** with the Propeller's I²C pins:

```
pub setup

  setupx(EE_SCL, EE_SDA)


pub setupx(sclpin, sdapin)

  longmove(@scl, @sclpin, 2)
  dira[scl] := 0
  outa[scl] := 0
  dira[sda] := 0
  outa[sda] := 0

  repeat 9
    dira[scl] := 1
    dira[scl] := 0
    if (ina[sda])
      quit
```

We begin by making a copy of the pins for use in other methods. For each pin, we write a 0 to the **dira** bit (which floats it to the pull-up), and then write 0 to the output bit. This last step is important as we'll only be manipulating the direction bit later; when we make a pin an output, we want the line to be pulled low. Hence, the 0 in the associated **outa** bit.

The next section is a trick I learned from Mike Green's object. This wiggles the clock line until the SDA pin is released by the slave. This is useful for clearing the state of the buss before doing anything else.

Transactions begin with the Start condition:

```
pub start

  dira[sda] := 0
  dira[scl] := 0
  repeat
  while (ina[scl] == 0)

  dira[sda] := 1
  dira[scl] := 1
```

Just to ensure we know the state of the pins, the **dira** bits for SDA and SCL are set to input mode to float the lines to the buss pull-ups. Before proceeding, the SCL pin is checked to ensure that it did, in fact, go high. A low on the SCL line can indicate a device on the buss is not ready (this is called "clock stretching"). As soon as SCL is clear (high), we take the SDA line low by writing a 1 to the SDA's **dira** bit. Remember, in the **setupx()** method, we had previously written a 0 to the **outa** bit. The next line takes the SCL line low in preparation for clock pulses used by **write()** and **read()**.

Let's have a look at the **write()** method:

```
pub write(i2cbyte) | ackbit

  i2cbyte := (i2cbyte ^ $FF) << 24
  repeat 8
    dira[sda] := i2cbyte <-= 1
    dira[scl] := 0
    dira[scl] := 1

  dira[sda] := 0
  dira[scl] := 0
  ackbit := ina[sda]
  dira[scl] := 1

  return ackbit
```

Honestly, this method is simpler than it appears at first blush — but it has generated more questions than nearly anything I've ever produced. Let's tackle it, line by line.

We begin by inverting the bits of our data byte using XOR (^) with $FF. Why? Because we're not going to be moving the bits from *i2cbyte* directly to the SDA pin; we're going to control the output using the **dira** bit for the pin. To output a 1 to the buss, we need to put a 0 in the **dira** bit. This puts the pin in input mode and allows the pull-up to take the SDA line high. Conversely, to write a 0 to the buss, we need to write a 1 to the **dira** bit to put it into output mode (again, we previously wrote 0 to the **outa** bit).

After flipping the bits, we shift everything left by 24. This moves bit7 (MSB) of *i2cbyte* to bit31. This is necessary because all parameters and local variables are longs.

A simple **repeat** loop takes care of moving the bits from *i2cbyte* to the SDA line. The trick employed here is the rotate left operator. We're using the assignment version to simplify the code. The output will be bit0 of **i2cbyte** after the rotate.

When the loop is complete, the SDA and SCL lines are floated; the latter process causes the slave to output the ACK/NAK bit. The master samples the SDA line into *ackbit*, then returns the SCL line low for the next byte.

The **read()** method is similarly constructed, though a little easier to get through:

```
pub read(ackbit) | i2cbyte

  dira[sda] := 0
```

```
    repeat 8
      dira[scl] := 0
      i2cbyte := (i2cbyte << 1) | ina[sda]
      dira[scl] := 1

    dira[sda] := !ackbit
    dira[scl] := 0
    dira[scl] := 1

    return (i2cbyte & $FF)
```

We start by writing 0 to the **dira** bit for SDA which puts it into input mode. A **repeat** loop takes the SCL line high, shifts *i2cbyte* left one bit, then copies the state of the SDA line to bit0 of *i2cbyte*. After the SCL sampling, the SCL line is taken back low.

Earlier, I mentioned that the receiving end of the transaction provides an ACK/NAK bit back to the sender — this is passed as a parameter to the **read()** method. You'll need to check the datasheet for the component for ACK/NAK bit usage.

In general, I've found that the receiver will provide a ACK after all but the last byte read. We'll see this shortly.

Finally, the **stop()** method terminates a transaction:

```
pub stop

    dira[sda] := 1
    dira[scl] := 0
    repeat
    until (ina[scl] == 1)

    dira[sda] := 0
```

Memory Like an Elephant

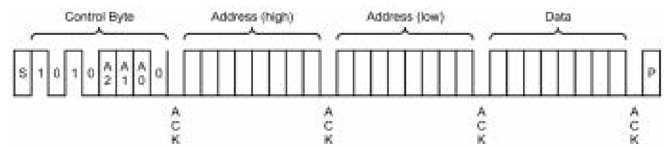
An EEPROM never forgets, and as we have one attached to every stand-alone Propeller project, we're going to use it as an I^2C training ground. If you don't have it, go download the datasheet for the 24LC256 or 24LC512 EEPROM (based on your board). You really want to have this handy.

Let's start by writing a single byte to the boot EEPROM. We'll need to do the following:

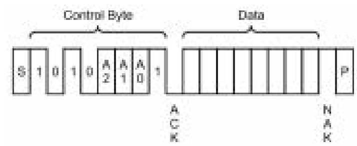
- Start condition
- Write control byte ($A0)
- Write high byte of address
- Write low byte of address
- Write data byte
- Stop condition

In **Figure 4**, you'll see a diagram that illustrates the transaction — this is what you'll find in most I^2C component datasheets.

Using the ***jm_i2c.spin*** object, we could add a method to our program like this:



■ FIGURE 4. EE write byte diagram.



■ FIGURE 5. EE read byte diagram.

```
pub wr_byte(ctrl, address, value)

    i2c.start
    i2c.write(ctrl & $FE)
    i2c.write(address.byte[1])
    i2c.write(address.byte[0])
    i2c.write(value)
    i2c.stop
```

Pretty simple, right? Note that we're AND-ing the control byte with $FE to make sure that bit0 is clear; this is for write mode.

Before I continue, let me point something out to those of you coming from the Arduino world, and to those that may want to port useful Arduino projects to the Propeller. The TWI (two-wire interface) library in the Arduino treats addresses differently that we're doing here. For example, if you look up Arduino code for an EEPROM, you'll see that the control byte for an EEPROM at address %000 is $50, not $A0 as we're using. The reason is that the TWI library takes the address provided and shifts it left by one bit to make room for the Read/Write bit in the control byte.

In my experience, most device vendors provide the control byte value as we're using here (not right-shifted), but you'll want to double-check that if you have problems connecting to a device. The transaction diagrams usually provide the clearest information vis-à-vis control byte.

Okay, let's read a byte back from the EEPROM. Since the EEPROM maintains an address pointer, the process can be this simple:

- Start condition
- Write control byte ($A1)
- Read data byte
- Stop condition

The transaction diagram for a simple byte read is shown in **Figure 5**. Here's the translation to Spin:

```
pub rd_byte | value

    i2c.start
```

```
    i2c.write(ctrl | $01)
    value := i2c.read(i2c#NAK)
    i2c.stop

    return value
```

Note that for reading, we have to set bit0 of the control byte; this is done by OR-ing with $01. The byte is read back and we send a NAK to tell the EEPROM this is the last byte to read.

This works but is not terribly practical because it's counting on the address pointer in the EEPROM to be in the right place. A better idea is a method that can read from any address. Here's the process:

- Start condition
- Write control byte ($A0 - write mode)
- Write high byte of address
- Write low byte of address
- Start condition
- Write control byte ($A1 - read mode)
- Read data byte
- Stop condition

Do you see what we did there? We use the write mode to move the EEPROM's address pointer, then we restarted the transaction in read mode to read the byte.

You should note that when we read or write a byte from the EEPROM, the internal address pointer is automatically incremented; this allows us to read blocks. That said, the EEPROM has page boundaries with the address that affect block writes and reads; when the address pointer hits the end of a page, it will wrap around to the beginning. This can be problematic if we're not paying attention. In the 24LC256 (32K) EEPROM, the pages are on 64-byte boundaries; in the 24LC512 (64K — what I tend to use), the pages are on 128-byte boundaries. We need to be aware of this when writing multi-byte values. In my EEPROM objects, I have a couple helper methods for this:

```
  pub page_num(addr)

    return (addr / PG_SIZE)


  pub page_ok(addr, len) | pg0, pg1

    pg0 := page_num(addr)
    pg1 := page_num(addr + len-1)

    return (pg1 == pg0)
```

The first converts a target address of a page number. The second will check to see that the length of what I want to write to an EEPROM will fit within the page of the starting address. This is especially useful if we're going to store message strings that could have variable lengths.

As I've touched on my EEPROM objects, let's look at practical examples of writing and reading values to the EEPROM. The approach is to use a master method for writing which wants an EEPROM address, the number of bytes to write, and a pointer (address of) to the block of bytes to write:

```
  pub wr_block(addr, n, p_src) | ackbit

    i2c.wait(devid)
    i2c.write(addr.byte[1])
    i2c.write(addr.byte[0])
    ackbit := i2c#ACK
    repeat n
      ackbit |= i2c.write(byte[p_src++])
    i2c.stop

    return ackbit
```

Right off the bat, you'll notice that I replaced the **i2c.start()** method with **i2c.wait()** — let's have a look at the latter:

```
  pub wait(ctrl) | ackbit

    repeat
      start
      ackbit := write(ctrl)
    until (ackbit == ACK)
```

This method generates a start condition, writes the control byte (*ctrl*), then checks the ACK/NAK bit. If we try to access the EEPROM while it's busy storing the last value we wrote to it, we'll get a NAK. As soon as the EEPROM is ready, we'll return from this method and continue. By using **wait()**, we don't have to pad our code with arbitrary delays.

Back to the **wr_block()** method ... after getting the go-ahead from the EEPROM via wait, we write the address (Big-Endian order) then drop into a loop that writes bytes from the hub, incrementing the hub pointer after each write. Note that we have a check value (*ackbit*) that was preset to ACK (0); if there's a problem with any of the writes, this will get set to 1. The calling code can check this to ensure all bytes were written without issue.

With the **wr_block()** method in place, we can create named methods for simple data types:

```
  pub wr_byte(addr, b)

    return wr_block(addr, 1, @b)
```


Jon "JonnyMac" McPhalen
jon@jonmcphalen.com

Parallax, Inc.
www.parallax.com
Propeller chips and programming tools

Propeller Activity Board	Parallax #32910

```
pub wr_word(addr, w)

  return wr_block(addr, 2, @w)


pub wr_long(addr, l)

  return wr_block(addr, 4, @l)
```

What about strings? You'll recall that the Propeller doesn't have a string type per se, but will treat an array of characters that is terminated with 0 as a string. We can, of course, use **wr_block()** with a string but we'd need to know its length for writing and reading. For this reason, my EEPROM object has separate support methods for strings:

```
pub wr_str(addr, p_zstr) | ackbit, b

  ackbit := i2c#ACK
  repeat
    b := byte[p_zstr++]
    ackbit |= wr_byte(addr++, b)
    if (b == 0)
      quit

  return ackbit
```

For **wr_str()**, we need to pass the EEPROM storage address and the hub address of the string. A loop is used to read a byte from the string and write it to the EEPROM. As the **wr_byte()** method does write the address, we don't have to worry about page boundaries with this method. The loop will terminate when it finds the 0 at the end of the string. The 0 was written to the EEPROM as well, so that the **rd_str()** method can detect the end of the string.

As with writing, we have a master method for reading bytes from the EEPROM:

```
pub rd_block(addr, n, p_dest)

  i2c.wait(devid)
  i2c.write(addr.byte[1])
  i2c.write(addr.byte[0])
  i2c.start
  i2c.write(devid | $01)
  repeat while (n > 1)
    byte[p_dest++] := i2c.read(i2c#ACK)
    --n
  byte[p_dest] := i2c.read(i2c#NAK)
  i2c.stop
```

As you can see, the setup is similar to **wr_block()**, but this time we provide a destination address (in the hub) for the byte(s) read from the EEPROM.

The loop used in this method is conditional. If the number of bytes left to read is greater than one, then the loop will run, reading a byte replying with ACK. When we're down to the last byte, the loop is skipped, and we do the read with a NAK reply. Easy peasy.

By now, you don't need me to spell out how the other read methods work. Open the EEPROM object and have a look. After you're done looking around, have a go at it. I know I tend to harp on this, but the best way to learn to write code is to write code!

Auto-Loading Values

From time to time, we'll create an application that requires the preservation of values between reset/power-up cycles. In Propeller projects, this is a particularly easy feature to add. How we approach it depends on an additional requirement: Do we want to preserve the values between new program downloads?

If the answer to the question is yes, then we need to use a 64K or larger EEPROM and write any saved values to address $8000 and higher. I did this for the EFX-TEK EZ-8+ controller which stores sequence data in the upper 32K of a 64K EEPROM.

If we can forgo saving data between program updates, there's a nifty little trick that we can take advantage of by storing a value in the EEPROM at the same address it uses in RAM. When the EEPROM is copied into RAM, variables and tables are populated — whatever was last written to the EEPROM will be loaded in the value.

For regular variables, the first value after a download will be 0. For many applications, I like to pre-set values to something other than 0, and this is where a DAT table is very helpful. As a reminder, a DAT table allows us to store values that look like constants in a program listing, yet behave like variables when the program runs. For example, in a game program I'm working on, there is a player table that has values like this:

```
dat { player information }

  PlayerID       byte   1
  WeaponType     byte   1
```

Once the game has been set up, we want to preserve these values between power cycles (e.g., battery change). Updating a table value in the EEPROM is as simple as this:

```
  ee.wr_byte(@PlayerID, PlayerID)
```

The **wr_byte()** method takes two parameters: the storage address and the byte value. Note that I'm using the address (@) of PlayerID to save the value of PlayerID.

Again, this process also works with regular variables. The difference is that we can initialize DAT table values in the program listing; regular variables cannot be initialized in a listing.

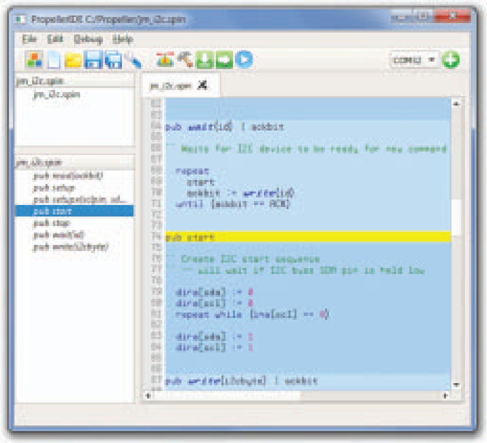


FIGURE 6. Propeller IDE.

Open Source Propeller IDE

While my friends at Parallax may have arrived late to the open source party, it's safe to say that they're fully onboard and supporting it now. One of the more exciting projects in the Propeller world is a new open source IDE for those of us that prefer Spin over C.

You probably know by now that there is a very nice cross-platform IDE for Propeller GCC. I — among many — asked if Spin could be folded into this tool (SimpleIDE). Well, it seemed like an innocent request at the time, but the ultimate implementation is not as clean as Spin programmers would like; the program was designed for the project-oriented nature of C.

While involved in another project, I asked — okay, cajoled — Steve Denson into taking that other project and turning it into a cross-platform, Spin-centric programming tool — something that should ultimately replace the Windows-only Propeller Tool. That project was built on a lot of the same code base as SimpleIDE, so it seemed like a workable idea.

In my view, Steve is one of the heroes of the Propeller community. It is through his efforts that really great cross-platform tools are becoming available for the Propeller. To be fair, Steve works with other great folks (in particular, Roy Eltham and Dennis Gately have been big contributors) that are likewise helping to create great new tools for our favorite microcontroller. When I made this twist-your-arm request of Steve, he said that he couldn't go it alone and then invited others to participate with him. This is the soul of open source, right?

It's still in progress, but looking very good — and it may well be in public release by the time you're reading this. **Figure 6** is a screenshot of the Propeller IDE running under Windows. As with SimpleIDE, this program uses the Qt framework and will run on Windows, Mac, and Linux.

Note the toolbar (which is still missing in the Propeller Tool). It's handy for mousers like me. Note, too, that the connection port to the Propeller is on that toolbar so we don't have to go looking for it. What makes me happiest is the pane in the lower left of the screen. This will display the methods included in the current object. I cannot count the times I've had to open object files to scan through the method names to find what I need. Not any more. They are right there on the screen.

Also in the works are features we find in advanced editors like hints and auto-completion. I really appreciate the efforts by Steve and the team for doing this. I love the background colors in Propeller Tool, but find other areas lacking — not the least of which are variants for other computer platforms.

If you've been holding off trying Spin because you don't have a tool for your favorite operating system, it's time to file that excuse in the trash bin and join the Propeller party!

Until next time, keep spinning and winning with the Propeller!

ELECTRONET

■ BY L. PAUL VERHAGE

A Thermometer for the Totable Thermal Vacuum Chamber (t-TVC)

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2014_NearSpace.**

I'm still in the process of upgrading my thermal vacuum chamber (TVC) design. Part of the current upgrade is incorporating a thermometer. The problem is that I haven't located an affordable thermometer that reads below -50°F. So, when a product is not available, we readers of *Nuts & Volts* go the DIY route and make one ourselves. This month, I'll discuss the design of my thermometer for TVCs.

Sending payloads into near space is far less expensive than sending an identical payload to low earth orbit (LEO). That doesn't mean, however, that near space is cheap. A near space launch costs about $120, putting the cost to around $10 per pound of payload. That compares favorably with launches to LEO where the cost is much closer to $10,000 per pound.

■ Here's a recording thermometer that operates down to -99°F. It's just the tool you'll need when simulating near space.

Even at a low $120 per flight, you want to increase your chances of a successful mission. This is why testing payloads prior to a mission is critically important for anyone contemplating a near space launch. One of these tests should be a cold soak in near space like conditions.

Conditions in Near Space

If you're a reader of this column, you'll know that near space is a bitterly cold vacuum. Temperatures drop to -60°F on a warm day and down to -90° during winter. There are a lot of things you might want to send into near space that don't like these conditions. Here's an example.

Batteries use a chemical reaction to create a current. Like all chemical reactions, a battery's reaction rates increase when its temperature is higher. Operating a payload with the wrong type of battery is an invitation to disaster. That's because once the battery's temperature gets too cold, payload power ceases and the experiment fails. So, where can you test a battery prior to a near space launch? In a thermal vacuum chamber (TVC).

Thermal Vacuum Chambers

I wrote a series of articles on TVCs when the availability of affordable Chinese vacuum pumps made them a realistic bench-top test tool of near space research. Essentially, the near space TVC I've written about previously is a metal can connected to a vacuum pump. The exterior of the can is surrounded by dry ice and the interior is evacuated of air. Inside, the item under test experiences the cold and vacuum of near space.

The issue I've always faced with my TVC design is the determination

of the temperature inside the chamber. In the past, I've relied on a data logger inside the chamber to collect and store the temperature at regular intervals. Then, after the test was over, I downloaded the stored data and graphed the results.

This means I didn't know the temperature of the test run until after it was completed. I've looked into thermometers for displaying the current temperature, but found that many didn't display temperatures as cold as I needed for a near space TVC. After sitting on the back burner for a few years, I decided to revisit this issue. This time, I created a solution.

A DIY Recording Thermometer for TVCs

Here's the design I finally settled on. I utilized a PICAXE microcontroller to measure the temperature and then display the results. This way, the thermometer can record the temperature and download the results for analysis after the test. (It just goes to show that adding a microcontroller to a DIY project increases the abilities of the final product and makes it much smarter.)

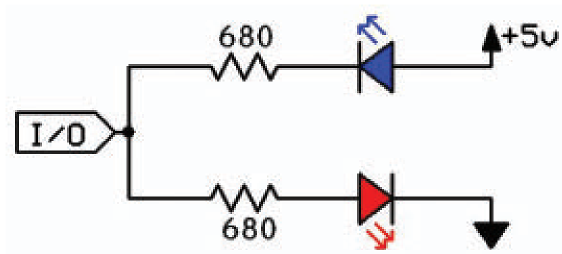
The initial design required that I create a printed circuit board (PCB) for two seven-segment LED displays and two BCD to seven-segment display decoders. This is where I discovered I would need to pay attention to whether the seven segments were common anode or common cathode.

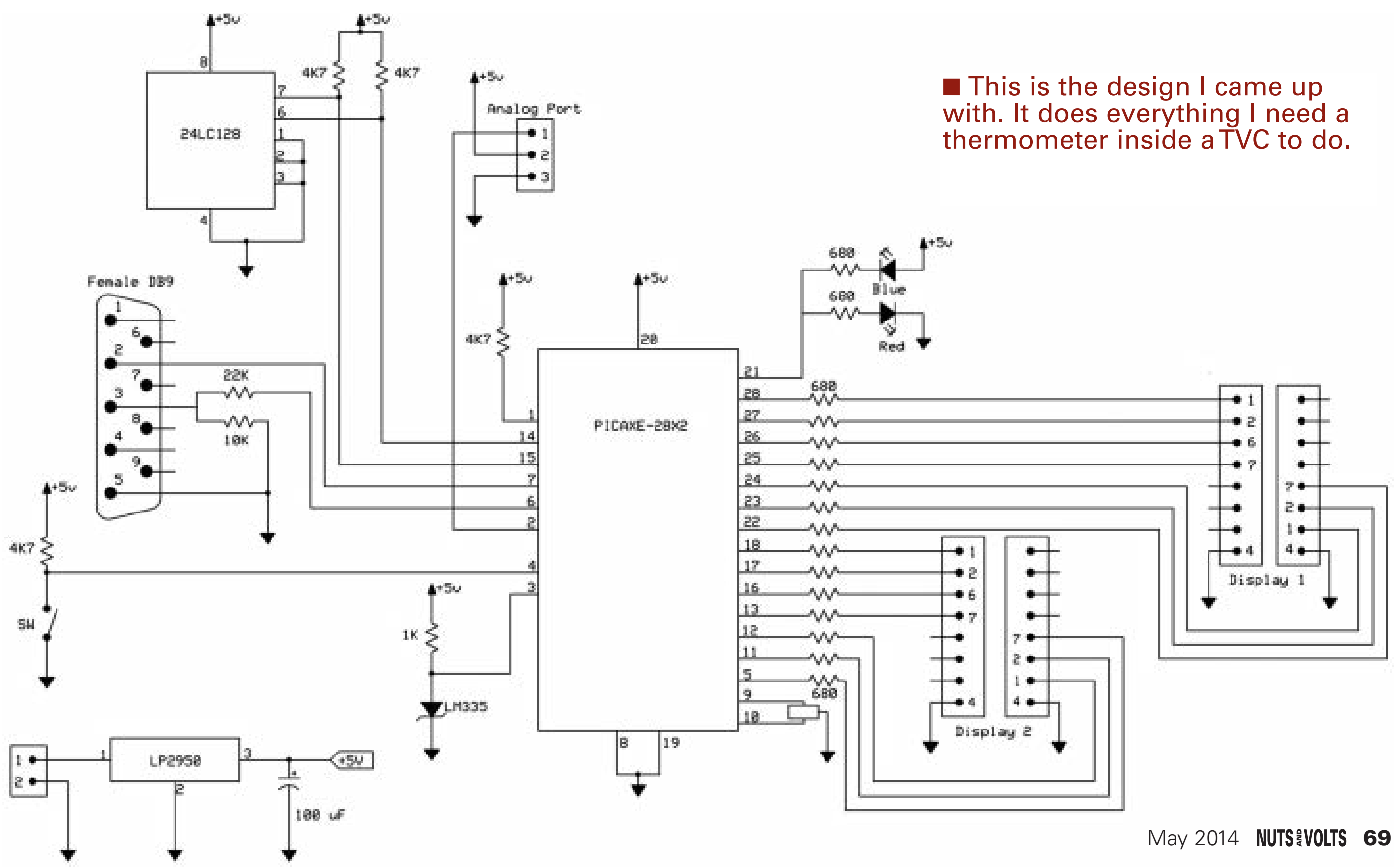
Since LEDs cannot be operated in reverse bias, there's no way to connect a common cathode display to a common anode decoder. Yet, this is the exact PCB I ended up sending to production.

After receiving two absolute failure PCBs, I decided I had enough trying to match decoders to displays. I would let the PICAXE do all the work, rather than pass off some of the work to the decoders. This meant I would need a larger PICAXE.

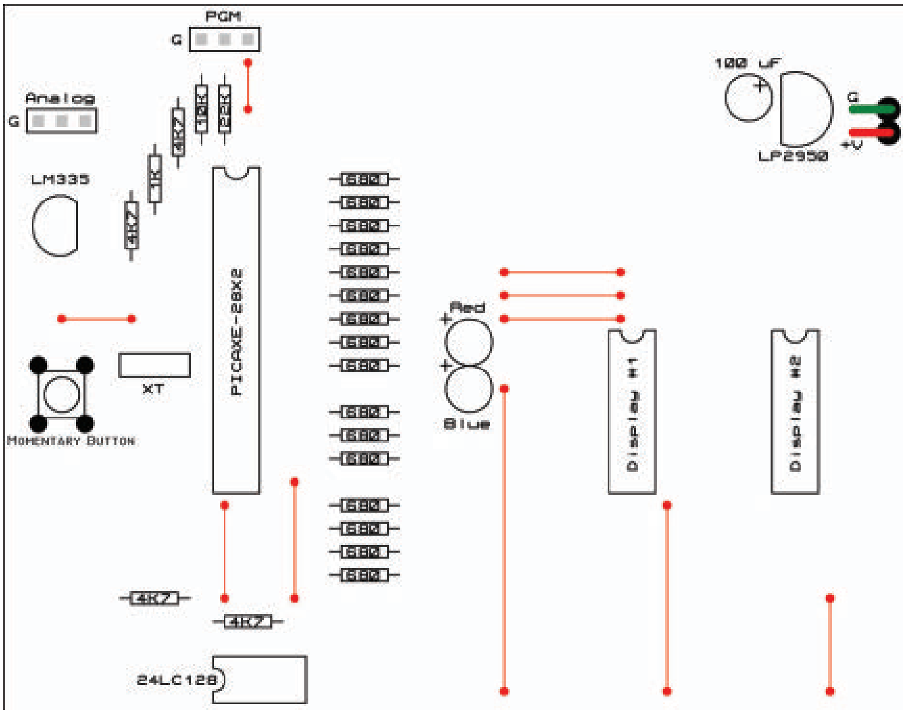
Driving long distances is a wonderful method for solving problems. While traveling on vacation earlier this year with my wife, I was able to mull over a design in my head and determine the minimum PICAXE I needed. It would have to operate a total of 14 LED segments, a bi-color LED for a sign indicator, measure the

■ This simple circuit lets a single microcontroller operate two LEDs. In the past, I used two I/O pins and a bi-color LED to generate the same effect. Be sure you add current-limiting resistors to both LEDs if you plan to use this circuit.





■ This is the design I came up with. It does everything I need a thermometer inside a TVC to do.



■ This is the placement of components for the copper mask. The red lines are jumper wires because the design is only single-sided. Use three-pin male headers for the analog and programming ports. XT is a 10 MHz resonator. The two LED displays fit in 14-pin DIPs and are found at **Jameco.com** as part number 24782. Voltage is any battery pack, as long as the voltage is 6.0 volts or greater.

voltage from a temperature sensor, and determine if it needed to record or download temperature data.

I concluded I needed a PICAXE-28X2 for the job. The PICAXE-28X1 had one too few outputs.

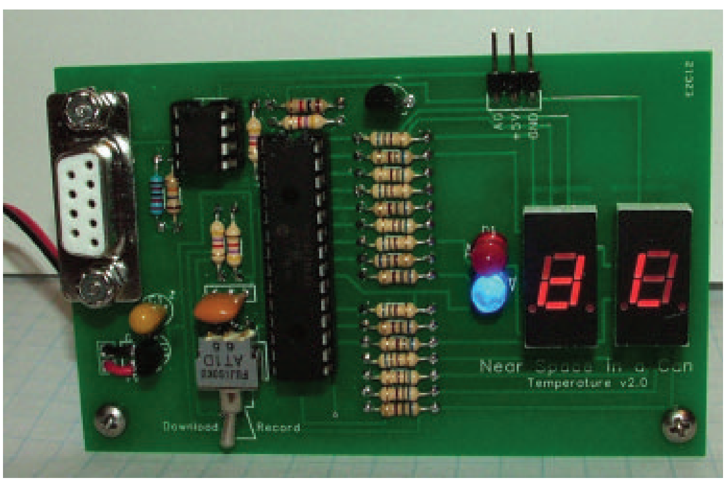
Before beginning the design process, I needed to perform an experiment. Can I use a single I/O pin to display two colors? This would be important since I wasn't planning to use a seven-segment LED display that included a sign (+ or -). How else could I display whether or not the temperature was above or below zero unless I used a bi-color LED?

The problem with using a bi-color LED was that I would be limited to displaying red or green, and would have to use two I/O pins to do so. Since a single I/O pin could have two states (high and low), I was banking on the fact that I could turn on one LED and the other off by toggling the state of the I/O pin. For this test, I developed the circuit in **Figure 2**, and discovered it worked exactly as I planned. (I was worried I could be overlooking a problem that would prevent this circuit from working.)

If the I/O pin is set high, (conventional) current flows from the I/O pin through the red LED and its resistor. There's no way for current to flow through the blue LED. When the I/O pin is set low, current flows from the five volt supply through the blue LED, and to ground via the I/O pin. There's no way for current to flow from the five volt supply through both LEDs and to ground as long as the I/O pin is set either high or low.

If the I/O pin is in a high impedance input state, then it is possible to illuminate both LEDs at the same time. However, this is not going to happen because the thermometer's program will set the I/O either high or low.

In the thermometer, the I/O pin is set high (illuminates the red LED) to indicate above zero temperatures and set low (illuminates the blue LED) to indicate below zero temperatures.



■ During a download, I programmed the thermometer to display dL. This way, I could confirm whether or not the thermometer knows its suppose to be dumping data.

Since the PICAXE could operate two different color LEDs with a single I/O pin, this created the opportunity to include a new feature in the microcontroller-based thermometer: It could record a voltage from the item under test.

Perhaps I'll add a pressure sensor to the circuit in the future, but since the TVC already has an analog pressure sensor, I'm in no hurry. **Figure 3** is the schematic of the thermometer design I came up with.

For readers who would like to try making this circuit on their

own, I created a bottom copper mask (available at the article link). Since this is for a single-sided PCB, the design isn't as compact as my professionally-made boards.

The Thermometer Program

There's a sample of the thermometer's program also online at the article link. Below is the fifty cent tour of the program.

The program begins by setting up I^2C communications with the 24LC128 EEPROM. The data will be stored here, and the PICAXE needs to communicate at 400 kHz and in one word addresses. Next, the code zeros out the LED displays to show 00.

Prior to measuring the temperature with the LM335, the PICAXE checks to see if the thermometer's switch is set to Run or Download. If set to Download, the PICAXE starts at the beginning of the memory's address and gets each value stored.

The data — which is stored in words (since the PICAXE uses the 10-bit A-to-D (analog-to-digital) command for greater temperature resolution) — is dumped onto a PC using the *SERTXD* command. This means the user must open the Terminal program under the PICAXE Editor.

In Run mode, the PICAXE uses the *READADC10* command to digitize the voltage produced by the LM335 temperature sensor. At 10-bit resolution, the thermometer program can measure changes as small as a single degree Fahrenheit. This also means the results must be stored in one word records (rather than smaller one byte records).

The math routine that converts digitized voltage readings into a two-digit display first determines if the temperature is above or below zero. This occurs at a digitized reading of 522 for the sensor I am using. You, however, might find you'll need to adjust this value up or down one number.

Every change in one digitized voltage reading (say, from 522 to 523) is a change by 1.16 degrees Fahrenheit. Therefore, for every change in seven digitized voltage readings, the temperature is decremented by one.

Whether the temperature is above or below a reading of 522 determines whether I/O pin B.0 is set high (to illuminate the red LED) or low (to illuminate the blue LED). The LED digits represent the tens and ones of the temperature reading. Each is set separately, so the program must divide the temperature value by 10 to separate out each digit.

There's a subroutine to show the digits zero through nine on each display, and the program jumps execution to the proper subroutine.

After displaying the temperature, the thermometer program records the digitized temperature reading into the next address in the 24LC128 memory. After a pause (you program the length of the pause), the program repeats itself.

That's about it for the Totable TVC (t-TVC) Thermometer. It's a fun project that's useful for applications off the work bench. We had some pretty cold days in Idaho this winter (although not as bad as the upper Midwest and East experienced), and I would have liked to use to record and display temps overnight. Perhaps you'll find this project equally interesting and useful.

For readers who don't want to make their own PCB, I'm making kits available on my website at **NearSys.com**.

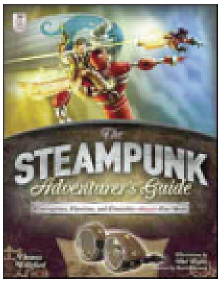
Onwards and Upwards,
Your near space guide **NV**

For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

GREAT FOR DIYers!

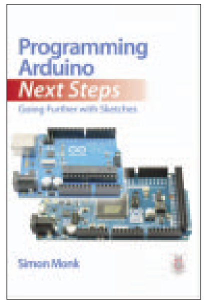
The Steampunk Adventurer's Guide
by Thomas Willeford



Steampunk stalwart Thomas Willeford cordially invites you on an adventure — one in which you get to build ingenious devices of your own! Lavishly illustrated by award-winning cartoonist Phil Foglio, *The Steampunk Adventurer's Guide: Contraptions, Creations, and Curiosities Anyone Can Make* presents 10 intriguing projects ideal for makers of all ages and skill levels, woven into an epic tale of mystery and pursuit.

$25.00

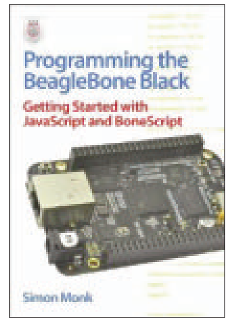
Programming Arduino Next Steps: Going Further with Sketches
by Simon Monk



Arduino guru Simon Monk reveals advanced programming techniques for Arduino! *Programming Arduino Next Steps: Going Further with Sketches* is the must-have follow-up to Monk's bestseller, *Programming Arduino: Getting Started with Sketches*. Aimed at experienced programmers and hobbyists who have mastered the basics, this book takes you "under the hood" of the Arduino, revealing professional-level programming secrets.

$20.00

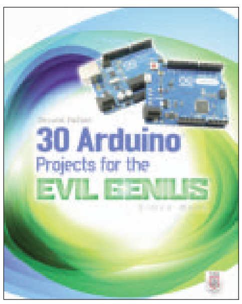
Programming the BeagleBone Black: Getting Started with JavaScript and BoneScript
by Simon Monk



Learn how to program the BeagleBone Black — the wildly popular single-board computer — using JavaScript and the native BoneScript language. You'll find out how to interface with expansion capes to add capabilities to the basic board, and how to create a Web interface for BBB. Two hardware projects demonstrate how to use the board as an embedded platform.

$15.00

30 Arduino Projects for the Evil Genius: Second Edition
by Simon Monk

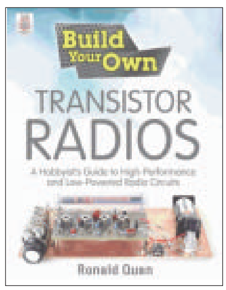


Fully updated throughout, this do-it-yourself guide shows you how to program and build fascinating projects with the Arduino Uno and Leonardo boards, and the Arduino 1.0 development environment. *30 Arduino Projects for the Evil Genius, Second Edition*, gets you started right away with the simplified C programming you need to know, and demonstrates how to take advantage of the latest Arduino capabilities.

$25.00

Build Your Own Transistor Radios
by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits

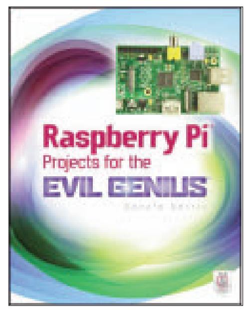


Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.

Paperback, 496 pages

$49.95

Raspberry Pi Projects for the Evil Genius
by Donald Norris

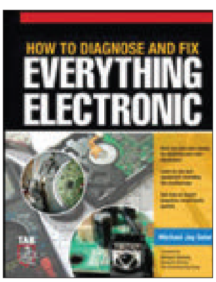


This wickedly inventive guide shows you how to create all kinds of entertaining and practical projects with the Raspberry Pi operating system and programming environment. Each fun, inexpensive Evil Genius project includes a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions for easy assembly. The larger workbook-style layout makes following the step-by-step instructions a breeze.

$25.00

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier

Master the Art of Electronics Repair



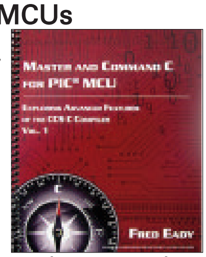
In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.

$24.95

Programming PICs in Basic
by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Master and Command C for PIC MCUs
by Fred Eady



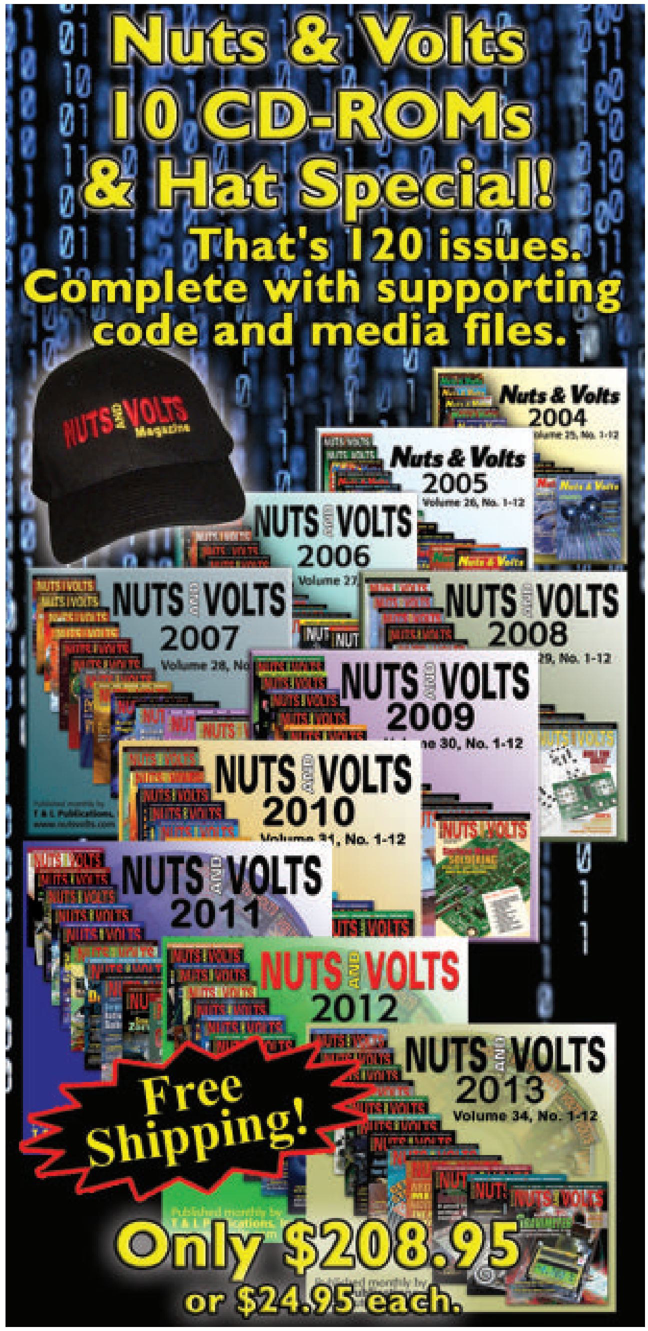
Master and Command C for PIC MCU, Volume 1 aims to help readers get the most out of the Custom Computer Services C compiler for PIC microcontrollers. The author describes some basic compiler operations that will help programmers particularly those new to the craft create solid code that lends itself to easy debugging and testing. As Eady notes in his preface, a single built-in CCS compiler call (output_bit) can serve as a basic aid to let programmers know about the "health" of their PIC code.

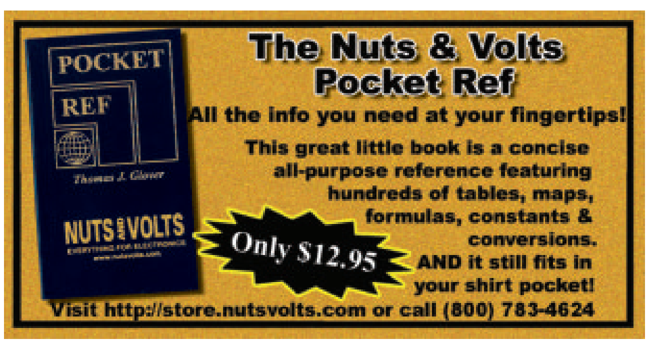
$14.95

Order online @ **www.store.nutsvolts.com**
Or CALL **1-800-783-4624** today!

BOOK & KIT COMBOS

CD-ROM SPECIAL

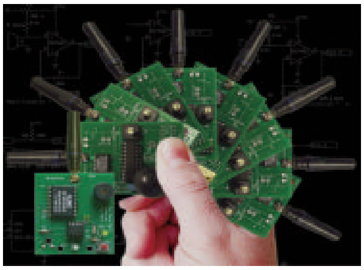




CALL 1-800-783-4624 today!
Order online @ www.nutsvolts.com

PROJECTS

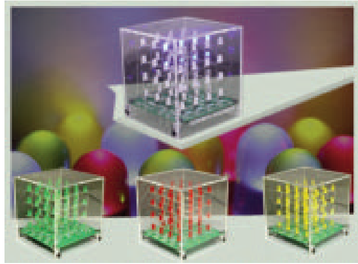
Super Detector Circuit Set



Pick a circuit!
With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.

$32.95

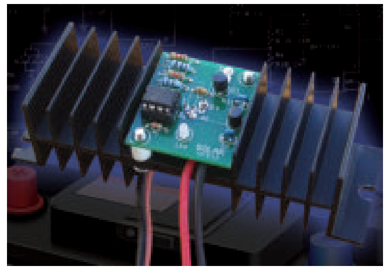
3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue

$57.95

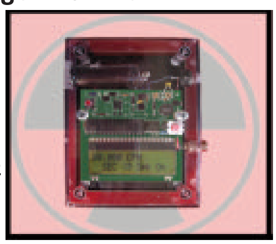
Solar Charge Controller Kit



This 12 volt/12 amp charge controller is very inexpensive and fits onto a heatsink that is mountable. Since the voltage is low, there is no danger of shock. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill. Protect your batteries from overcharging!

$23.45

Geiger Counter Kit



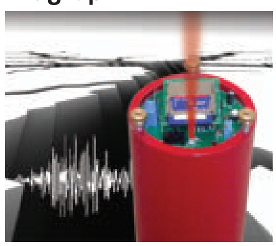
As seen in the March 2013 issue.

This kit is a great project for high school and university students. The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

Partial kits also available.

$145.95

Seismograph Kit



As seen in the May 2012 issue.

Now you can record your own shaking, rattling, and rolling.

The Poor Man's Seismograph is a great project /device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

$79.95

Battery Marvel

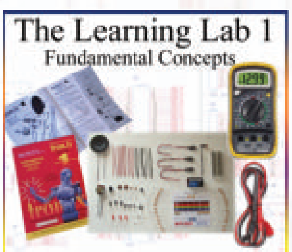
As seen in the November 2011 issue.

Battery Marvel helps protect cars, trucks, motorcycles, boats, and any other 12V vehicles from sudden battery failure. This assembled unit features a single LED that glows green, yellow, or red, indicating battery health at a glance. An extra-loud piezo driver alerts you to any problems.

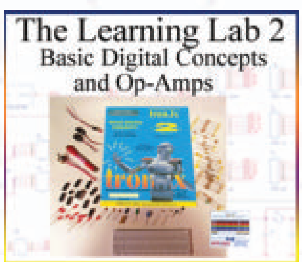
Details, please visit our website.

$32.95

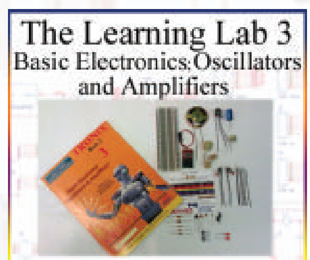
FOR BEGINNER GEEKS!



$59.95



$49.95



$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

CLASSIFIEDS

SURPLUS

SURPLUS ELECTRONIC PARTS & ACCESSORIES

Over 20,000 Items in Stock.

Cables, Connectors, Displays, Fans, Hardware, LEDs, Motors, Potentiometers, Relays, Semiconductors, Service Manuals, Speakers, Switches, Test Equipment, Tools, VCR Parts

Surplus Material Components
SMC ELECTRONICS
www.smcelectronics.com

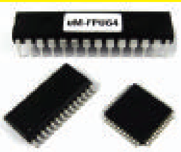
No Minimum Order.
Credit Cards and PAYPAL Accepted.
Flat $4.95 per order USA Shipping.

COMPONENTS

RF PARTS™ CO.

Tubes, Transistors, Power Components
Email: rfp@rfparts.com • Web: www.rfparts.com
800-737-2787 Fax 888-744-1943

uM-FPU64
64-bit Floating Point Coprocessor



64-bit and 32-bit IEEE 754 compatible
SPI or I2C interface, 3.3V Supply, 5V Tolerant I/O
Extensive floating point support, FFT, matrix operations
GPS input, local device support, Flash functions, IDE

Robotics
Navigation
Sensor Modules
Embedded Systems

Additional products...
uM-FPU V3.1 32-bit FPU
uM-PWM1 Servo Coprocessor

DIP-28, SOIC-28,TQFP-44

www.micromegacorp.com

HARDWARE WANTED

DEC EQUIPMENT WANTED!!!
Digital Equipment Corp.
and compatibles.
Buy - Sell - Trade
CALL KEYWAYS 937-847-2300
or email buyer@keyways.com

SERVICES

Sandtronics
Sandpiper Electronics Services LLC
"design of custom electronics"
Prototype electronics development for research, industry & inventors. Experience in laser systems, flight experiments & laboratory research applications.
www.sandtronics.com

Nuts & Volts Mobile App!
Check it out!

MOBILE SCOOTERS

Sunny Solutions
Sales and Service
Need Help Getting Around?
Elec Mobile Scooters
www.ScootersByTom.com
800 315-0309

KITS/PLANS

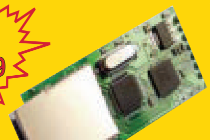
QKITS LTD
sales@qkits.com
1 888 GO 4 KITS
Arduino • Raspberry Pi
Power Supplies
MG Chemicals
RFID

Visit us at:
www.qkits.com

ELECT DATA SYSTEMS

S6000:TTL to Ethernet Converter

Only $17.99

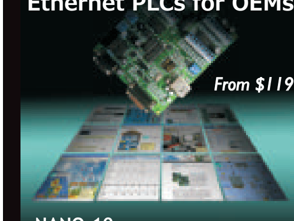


- Auto detect 10/100M High speed Ethernet
- Baud rate 300 ~ 25600 bps
- Agreement: ETHERNET, ARP, IP, UDP, TCP
- Can work as Virtual COM
- 3.3V or 5V power input interface

Ethernet,Zigbee,Rs485 IO modules at
WWW.SHJELECTRONIC.COM

CONTROLLERS

INTEGRATED
Ethernet PLCs for OEMs



From $119

NANO-10
- ETHERNET / Modbus TCP/IP
- 8 DI/Os, 2 AI/Os
- RS485
- PWM / PID / Stepper Control
- Ladder + BASIC Programming

tel : 1 877 TRI-PLCS
web : www.triplc.com/nv.htm

TRIANGLE RESEARCH INTERNATIONAL

AUDIO/VIDEO PARTS

ICEpower MODULES
50W x 2 (50ASX2) or
170W x 1 (50ASX2BTL)
Just add AC & speaker(s)
$99 USA $119 Worldwide
www.classicaudioparts.com

CLEARANCE SALE

HUGE CLOSE-OUT SALE!
All kinds of electronic products and parts!
Test equipment, schematics, tubes, audio-video equipment ... too numerous to list. Call for availability.
loustvelectronics@verizon.net
215.271.1138 or 215.468.3028
215.432.0333
Call Anytime/24-7

LIGHTING

ROBOTICS

MaxBotix®
High Performance Ultrasonic Rangefinders

Check us out Today!

Save 10% NOW On Web Orders
Enter Code: NV140531

HRXL-MaxSonar®-WR™
- High noise tolerance
- IP67 rated
- 1 mm resolution
- Multi-Sensor operation
- Calibrated beam pattern
- Starting at $109.95

XL-MaxSonar®-EZ™
- Great for UAV's and robotics
- Incredible noise immunity
- Small in size
- 1cm resolution
- Automatic calibration
- Starting at $39.95

Phone: 218-454-0766 Email: sales@maxbotix.com
www.maxbotix.com

Did You Know **Preferred Subscribers get access to all the digital back issues of *Nuts & Volts* for free?**

Call for details
1-877-525-2539
or go to
www.nutsvolts.com/faqs

Got bot?
www.servomagazine.com

READER-TO-READER TECHFORUM

All New TECHFORUM Online At www.nutsvolts.com/tech-forum

>>> QUESTIONS

3D Printer Materials Reference

I'm designing an RC dune buggy using a 3D printer, but I can't decide which material to use. I want it to be as light and strong as possible, but I need a reference that shows (for example) the strength of an I-beam made of ABS vs PLA. Does anyone know of such an engineering reference?

#5141 **Peter Murphy**
Pensacola, FL

Thermocouple Switcher

I want to read and log temperatures from multiple points in my (unnecessarily complicated) central heating system. I want to use thermocouples because: **1)** they are easy to attach to my hot water cylinder (stabbed under the insulation); and **2)** I already have them — from IKEA meat thermometers.

I would like to select each thermocouple in turn, use a single amplifier /analog-to-digital stage (Max31855 ??) to obtain the reading, and move on to the next thermocouple.

I need ideas for switching between the different thermocouples without introducing unwanted switch resistances, etc. Would an array of reed switches driven by a shift register be a sensible solution? Is there a better solid-state solution?

#5142 **Mike O'Hagan**
Aberdeen, Scotland, UK

Dimmable LED Bulbs

I've seen replacement LED light bulbs in the hardware stores that claim they are 'dimmable.' Is this a real feature that has a different internal design or is it just a way to get me to pay extra for an LED bulb? If there is a difference, what are they doing circuit-wise to make them dimmable?

#5143 **Robert Parsons**
Detroit, MI

What Wall Wart?

While on vacation, I managed to lose the power supply wall wart to a cheap vintage portable no-name brand shortwave radio. On the back of the radio, the power jack says 9 VDC and has a C shaped circle and a dot in the center with a plus sign on the dot. I'm pretty sure this means nine volts; positive tip. However, what it doesn't state is the milliamp rating. If I use a power supply that has too high or low an amperage rating, am I in danger of damaging the radio? How can I determine the right supply?

#5144 **Lyle Gardner**
Vista, CA

Pot Cleaning

I recently purchased a small "pig-nose" guitar amplifier at a neighborhood garage sale. I put batteries in it and it works, but the volume knob is very "scratchy" and at certain places in its rotation, the sound cuts out entirely. Is this something I could fix by cleaning the potentiometer and if so, what would I use? If it's time for a replacement, does anyone know where I could find a schematic for this unit?

#5145 **Emanuel Estrada**
Denver, CO

>>> ANSWERS

[#3141 - March 2014]

Motor Kickback

I am using an Ametek 965922-101 brushed DC motor (rated 38V nominal, 12A peak) in a project that runs it as both a drive and a brake. (Interestingly, the 38V rating is a "bus rating," i.e., with the motor stalled at 38V, it will use 12A and get nothing done.) In drive mode, the controller typically runs it up to 60V; current limited to 10A. Works great.

*For the brake, I am using an IRFB260 MOSFET with a 0.6 ohm spring in series to limit the surge current. The PWM rate is 120 Hz. (**NOTE:** When sinking 1/2 HP of energy, the motor generates 30V.) However, each time the MOSFET turns off (at 120 Hz), the motor provides a huge kickback, and the MOSFET's reverse-protection zener diode probably will not survive that for very long.*

At first, I paralleled the motor with an Elite 100 µF 400V capacitor [marked PM 105°C, (M 9305)] which calmed down the splash nicely. Then, one day after some generous usage, a huge cloud of smoke boiled out of a severely melted capacitor. It was a very abrupt introduction to the (unmarked) ripple current rating on capacitors. At the moment, I'm just using the same 4,700 µF 100V bolt-mounted huge capacitor that filters the drive circuit. (When using the 100 µF capacitor, one of the relay contacts would switch the 4,700 µF capacitor out of the circuit.) However, the 0.64 ohm brake circuit discharges the capacitor so quickly

Send all questions and answers by email to **forum@nutsvolts.com** or via the online form at **www.nutsvolts.com/tech-forum**

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

(at 120 Hz) that brake control is either none or a lot.

The challenges are that the kickback is positive, it needs to be kept under 200V, and there is a lot of energy behind the kick from this motor. I ended up with 100 µF in the first place because a large (non-electrolytic) 10 µF capacitor did very little to the splash, which almost immediately would go beyond the 250V rating of the oscilloscope. In my parts bin are some large diodes such as DTV32, STPR1020CT (2), STPR2045CT (2), TYN058, and others, if that helps. I've read a little about snubber circuits, but would like some technical advice.

#1 If I assume that RLY1 is not connected to short out the power supply and is shown de-energized, when RLY1 is energized, the motor connections will be reversed and the motor (acting as a generator) will try to drive the positive rail of the supply more positive. C4 will soak up some of the reverse current and when Q1 is turned on, it will limit the voltage to about 12 volts (1 ohm x 12 amps). Input to the power supply has to be turned off at this time.

There are two problems: During the transition time of RLY1 when the motor is not connected to anything, the voltage will shoot to the moon, possibly damaging the motor, and the delay time in PCB17 may not turn on Q1 in time to limit the voltage below 200 volts. One solution is to connect a 12 amp diode and series resistor across the motor to limit the voltage to 200 volts (R = 200V/12A = 16 ohms). A 75 watt light bulb would no doubt work as the resistor.

Russell Kincaid
Milford, NH

#2 Quickly braking an electric motor requires a circuit that will quickly dissipate quite a bit of energy. Texas Instruments has produced a paper, "The Art of Stopping a Motor," available at **http://e2e.ti.com/blogs_/b/motordrivecontrol/archive/2013/10/18/the-art-of-stopping-a-motor.aspx**. TI also has motor control dev kits that offer braking circuits and techniques that might help in Tsidqah's circuit. The kit documentation usually includes code and circuits. The Microchip Technology application note, AN-905, "Brushed DC Motor Fundamentals," offers a way to provide a shunt to ground that brakes a motor: **http://ww1.microchip.com/downloads/en/AppNotes/00905B.pdf**. I suggest the designer replace the 4PDT relay with a MOSFET H-bridge circuit that offers more flexibility for various braking applications. Also, real time voltage and current measurements taken during braking with a load resistance should help determine the characteristics needed in a braking load and switching circuits. As for a testing load, consider a non-electronic clothes iron or a toaster oven.

Jon Titus
Herriman, UT

#3 I'm not sure what you are doing with the 4PDT relay as shown, so maybe it is a drawing error. The way it is drawn, the motor feed and motor are both shorted! For motor reversal, you only need a 2PDT relay. If you are paralleling contacts for increased current, that's a bad idea since there is no guarantee the contacts will open or close in unison. So, in effect, one contact will still carry the load. Paralleling for redundency is also not good because if a contact welds itself, bad things can happen when the parallel contact switches.

I assume — since there is no logic flow provided — that the TRIG signal goes low before the EBRK signal goes high (and what about some protection for the input to the opto-isolator U6?). Likely, the spike is exceeding the LED PRV rating. I would think clamping diode(s) would help with the CEMF and perhaps an R/C, as well. Hope this helps!

Len Powell
Finksburg, MD

[#3142 - March 2014]
Capacitor Forming

I pulled some excellent quality electrolytic capacitors from the power supply of an amplifier. The capacitors are rated at 40 VDC but were used in a 10V circuit. I wanted to use the capacitors in a 24 VDC circuit, but I was told that the capacitors "formed" at 10V and wouldn't work at 24 VDC, regardless of the original rating. Is this true?

#1 True, aluminum capacitors will deform with years of operation at reduced voltage (or no voltage). The typical aluminum capacitor will have a leakage current that increases with capacitance and voltage. A 40 µF 40V cap would have leakage in the order of 2 µA.

To test your capacitor, connect it through a 10K resistor to 24 volts (or 40 volts if it is available); if the voltage rises to 24 volts or more, it is okay. If not, leave it connected until the voltage does rise above 24 volts. You can use a smaller resistor to speed up the process, but don't let the cap get hotter than 50 degrees C.

Russell Kincaid
Milford, NH

#2 Short answer... maybe.

You can often reform electrolytic capacitors by slowly ramping the voltage up — or above — your expected voltage. Keep a current meter in series and measure the charge current (after you have reached your top voltage). It should be in the microamp to low milliamp range for a good capacitor. If it's higher than a milliamp or two, you have a cap that

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

is breaking down (high leakage). Remember, low voltage electrolytic caps are pretty cheap, so don't get caught by false economy.

Perry Ogletree
Murfreesboro, TN

[#4141 - April 2014]
555 Anomaly

I have found an interesting anomaly with the 555 IC. Everywhere I've read and looked, the #4 (reset) pin has voltage held high to allow the IC to conduct, and then dropped low to turn off (reset) the IC.

I have 8.58V at 39.5 mA at VCC (#8) pin when the IC conducts. As you can see in the ***schematic****, I am controlling the IC through the reset pin (#4) with a CdS photocell photoresistor. When light is removed from the CdS photocell and the IC conducts, it sends 6.5V at 18.2 mA to the output (#3) pin and LEDs. Apply light to the CdS photocell and the IC stops conducting; you then get -0.2V at 0.0 mA at the output (#3) pin. Since I added a 4.7 µF electrolytic capacitor to the output pin, the IC now conducts 7V to the LEDs.*

I am an electronics hobbyist who has been learning electronics for one year. I welcome any comments and ideas anyone has about what is going on with the IC.

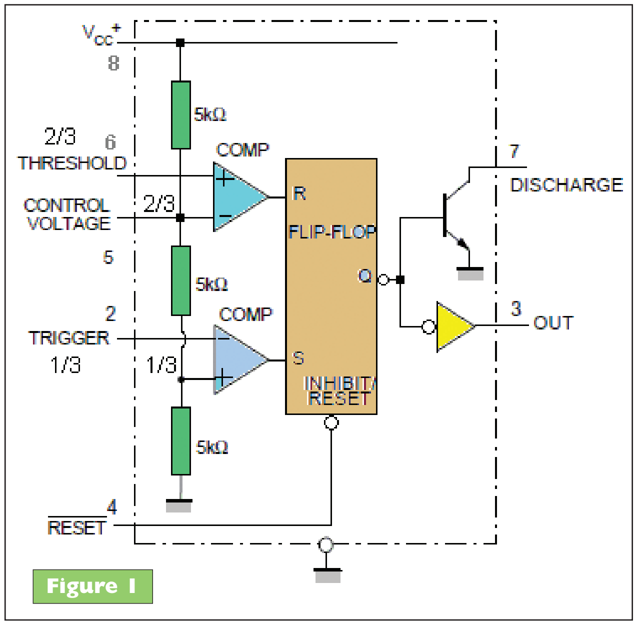


Figure 1

Your description of the behavior of your 555 circuit does not suggest anomalies but rather — as an old engineer once told me — "the circuit will always work the way you wire it." The 4.7 µF capacitor across the output terminal is putting an abnormal stress on the pullup circuit within the device. Voltage observed on pin 4 (/RESET) is normal for open-circuit conditions.

In order to use the 555 effectively, you need to know what's inside the device and how it works. When I design a 555 into a circuit, I always generate a schematic symbol that is more than just a rectangle with eight named pins. Rather, I show what's inside it — an equivalent circuit — because several years later I'll probably not remember exactly how one works and the added detail helps. **Figure 1** shows the equivalent circuit of a 555 and **Figure 2** shows its internal circuitry. Look at Q21 and

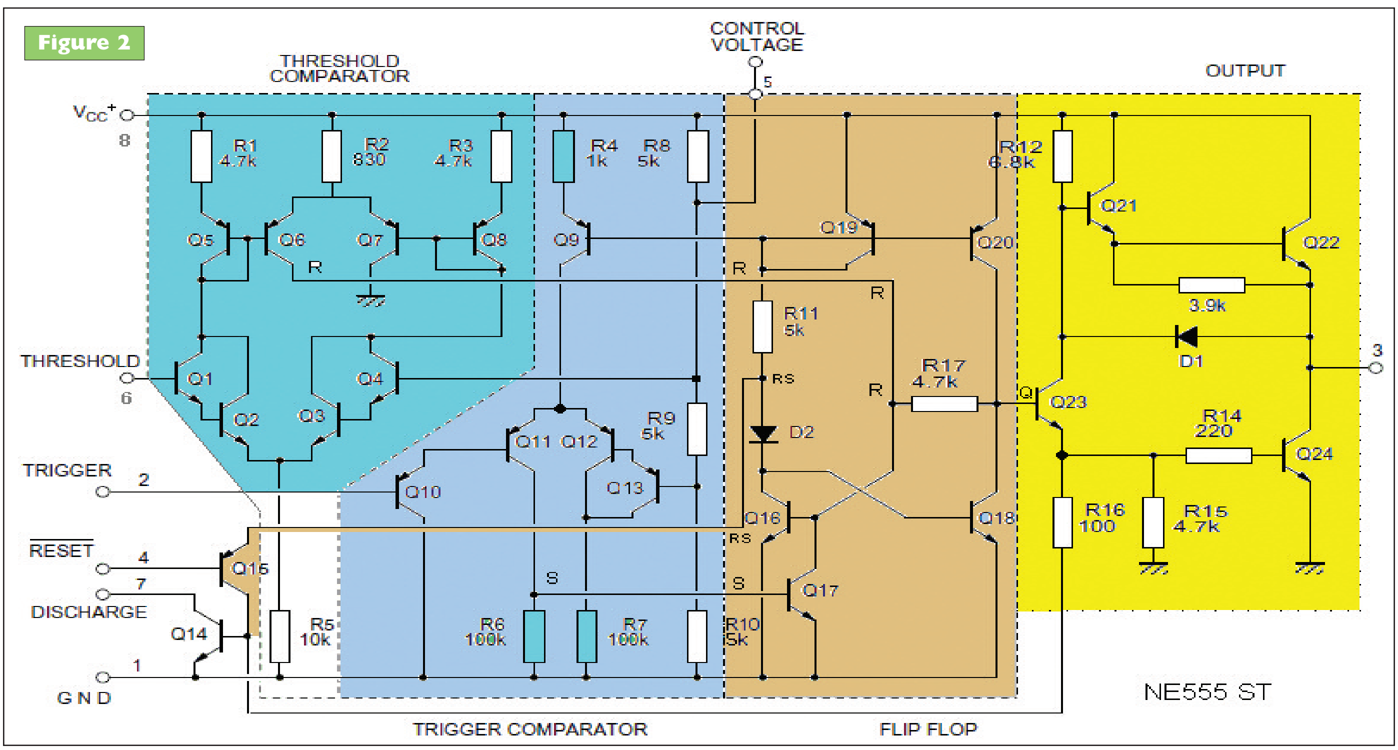


Figure 2

Q22 in **Figure 2** for some insight into what the 4.7 µF load might be placing on the circuit.

In the equivalent circuit, you'll see that the output is driven by a set/reset (SR) flip-flop which also has an overriding active-low master reset input (that is, you must pull the /RESET input to ground in order to reset the device). The Set (S) and Reset (R) inputs to the flip-flop are driven by comparators which are biased to 2/3 Vcc and 1/3 Vcc, respectively. If Vcc = 12 volts, then 1/3 Vcc = 4 volts and 2/3 Vcc = 8 volts. If the THRESHOLD input is made greater than 2/3 Vcc, the flip-flop will be reset and OUT (pin 3) will fall (to ground). If the TRIGGER input is made less than 1/3 Vcc, the flip-flop will be set and the OUT terminal will rise to Vcc.

In your application, the circuitry controlling the 555 can be simplified. Remove all of the components. Connect pins 2 (TRIGGER) and 6 (THRESHOLD) together. Connect this 2+6 pair through a pullup resistor to Vcc. Connect the 2+6 pair through your CdS photoresistor to ground. Choose the pullup resistor value such that a "dark" condition makes pins 2+6 rise to greater than 2/3 Vcc, and such that the "light" condition makes them fall to less than 1/3 Vcc. If you want the opposite behavior, reverse the circuit. Pull pins 2+6 to ground through the resistor and connect the fixed end of the CdS photoresistor to Vcc.

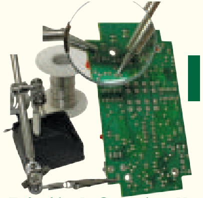
Finally, I caution you about paralleling LEDs. There is nothing to guarantee that they will all operate at precisely the same terminal voltage. Proper usage dictates that each has its own current-limiting resistor, and then you wire the eight LED+resistor pairs in parallel.

There are many, many articles on the Internet regarding 555 circuits, as well as application notes from Signetics, Phillips, and Texas Instruments. I recommend them to you. Good luck.

Peter A. Goodwin
via email

ADvertiser INDEX

Maker Faire®
YEAR OF THE MAKER
CELEBRATING COMMUNITY AROUND THE WORLD
BAY AREA
DETROIT
NEW YORK
KANSAS CITY
OSLO
UK
PARIS
ROME
GREATEST SHOW & TELL ON EARTH
TOKYO
SHENZHEN
9th Annual BAY AREA
MAY 17+18
makerfaire.com Brought to you by MAKE magazine
GOLDSMITH SPONSORS
intel
radioshack
SILVERSMITH SPONSORS
AUTODESK
Cognizant
MakerBot
ROBLOX



It's Time To Build A Kit!

We Put The FUN In Electronics!

Digital Controlled FM Stereo Transmitters

- ✔ PLL synthesized for drift free operation
- ✔ Front panel digital control and display of all settings and parameters!
- ✔ Professional metal case for noise-free operation
- ✔ EMI filtering on audio and power inputs
- ✔ Super audio quality, rivals commercial broadcasts
- ✔ Available in domestic kit or factory assembled export versions

Our Most Popular Transmitter... On Sale NOW!

For more than a decade we've been the leader in hobbyist FM radio transmitters. We told our engineers we wanted a new technology transmitter that would provide FM100 series quality without the advanced mixer features. They took it as a challenge and designed not one, but TWO transmitters!

The FM30B is designed using through-hole technology and components and is available only as a do-it-yourself kit with a 25mW output very similar to our FM25 series. Then the engineers redesigned their brand-new design using surface mount technology (SMT) for a very special factory assembled and tested FM35BWT version with 1W output for our export only market!

All settings can be changed without taking the cover off! Enter the setup mode from the front panel and step through the menu to make all of your adjustments. A two line LCD display shows you all the settings! In addition to the LCD display, a front panel LED indicates PLL lock so you know you are transmitting.

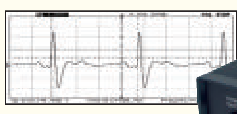
Besides frequency selection, front panel control and display gives you 256 steps of audio volume (left and right combined) as well as RF output power. A separate balance setting compensates for left/right differences in audio level. In addition to settings, the LCD display shows you "Quality of Signal" to help you set your levels for optimum sound quality. And of course, all settings are stored in non-volatile memory for future use! Both the FM30B and FM35BWT operate on 13.8 to 16VDC and include a 15VDC plug-in power supply. The stylish black metal case measures 5.55"W x 6.45"D x 1.5"H.

Take advantage of our ***Spring Sale***, and get our FMA200E 3.4dB gain FM antenna below to maximize your system while also saving big bucks!

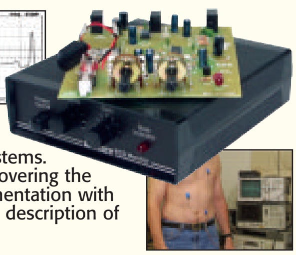
(Note: After assembly of this do-it-yourself hobby kit, the user is responsible for complying with all FCC rules & regulations within the US, or any regulations of their respective governing body. FM35BWT is for export use and can only be shipped to locations outside the continental US or valid APO/FPO addresses or valid customs brokers for end delivery outside the continental US.)

FM30B	**Digital FM Stereo Transmitter Kit, 0-25mW, Black**	SPRING SALE!	***$169.95***
FM35BWT	**Digital FM Stereo Transmitter, Assembled, 0-1W, Black (Export ONLY)**		***$259.95***

Electrocardiogram ECG Heart Monitor



- ✔ Visible and audible display of your heart rhythm!
- ✔ Bright LED "Beat" indicator for easy viewing!
- ✔ Re-usable hospital grade sensors included!
- ✔ Monitor output for professional scope display
- ✔ Simple and safe 9V battery operation



Use the ECG1C to astound your physician with your knowledge of ECG/EKG systems. Enjoy learning about the inner workings of the heart while, at the same time, covering the stage-by-stage electronic circuit theory used in the kit to monitor it. The documentation with the ECG1C covers everything from the circuit description of the kit to the circuit description of the heart!

Multiple "beat" indicators include a bright front panel LED that flashes with the actions of the heart along with an adjustable level audio speaker output that supports both mono and stereo hook-ups. In addition, a monitor output is provided to connect to any standard oscilloscope to view the traditional style ECG/EKG waveforms just like you see on ER... or in the ER! 10 hospital grade re-usable probe patches are included together with the matching custom case set shown. Safe 9V battery operation.

ECG1C	**Electrocardiogram Heart Monitor Kit With Case & Patches**	**$44.95**
ECG1WT	**Electrocardiogram Heart Monitor, Factory Assembled & Tested**	**$89.95**
ECGP10	**Electrocardiogram Re-Usable Probe Patches, 10-Pack**	**$4.95**

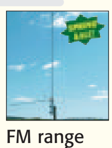
Passive Aircraft Monitor

PATENTED!

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1 Passive Aircraft Receiver Kit $89.95

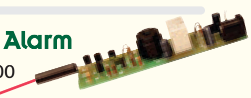
Collinear Vertical FM Antenna



Our 5/8 wave omni antenna has been the standard for LPFM installations worldwide. Provides 3.4dB gain while keeping the signal radiation low to the horizon for maximum range. Field tuneable over the entire FM range for a perfect match. SO239 connector for PL259 plug.

FMA200E Omnidirectional FM Antenna SPRING SALE! ***$119.95***

Laser Trip Sensor Alarm



True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1 Laser Trip Sensor Alarm Kit $29.95

Logic Interface Module

Interface your digital output to the real world with an on-board SPDT relay rated at 240V at 10A! It takes a digital low (.5VDC or less) or a high (+1 to +12VDC) and provides your choice of an active low or high closure! It's that simple! Runs on 12VDC at 60mA.

RI1 Logic Interface Kit $17.95

Four-Mode Keyless Entry Test Set

Just like the days of "plugs, points, and condenser" are over, so are the days of having the hardware store grind out a spare key for your car!

Now when your keyless access system doesn't work, you need to accurately detect what part of the system is malfunctioning. This could be anything from a dead battery in the key fob, a "brain-dead" key fob, to malfunctioning sensors, antennas, or other system components in the vehicle. Until now there was no way to determine where the system was failing.

Testing your system is easy. To test the complete 125 kHz/315 MHz communications path just stand close to the vehicle with the WCT3 and your key fob in hand. Press the test button and the WCT3 will detect and display the presence of the vehicle's 125kHz/20KHz signal and, if they "handshake", will also detect and display the presence of your key fob's 315MHz return signal. You can independently test key fob only signals (panic, lock, trunk, etc.) by holding the key fob near the WCT3, pressing the test button, and pushing the function button on the key fob.

The same functionality testing can be done with IR key fobs. The modulated IR signal is detected and will illuminate the IR test LED on the test set.

If you know a few "secrets" you can also see if the tire pressure sensors/transmitters are generating signals or the built-in garage door opener in your rear view mirror is transmitting a signal! Runs on a standard 9V battery. Also available factory assembled & tested.

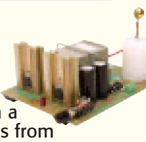
WCT3 Keyless Entry Test Set Kit $59.95

Air Blasting Ion Generator

Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 Ion Generator Kit $64.95

HV Plasma Generator



Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

PG13 HV Plasma Generator Kit $64.95

Speedy Speed Radar Gun

Our famous *Speedy* radar gun teaches you doppler effect the fun way! Digital readout displays in MPH, KPH, or FPS. You supply two coffee cans! Runs on 12VDC or our AC121 supply.

SG7 Speed Radar Gun Kit $74.95

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.

PR2 Broadband RF Preamp $69.95

Van de Graaff Generators

Create your own lightning with these time tested student favorites! Produces low current static currents that can be "shocking" but perfectly safe! Draw sparks to a screwdriver, grab hold and watch your hair stand straight up! Two models produce from 200kV to 400kV!

VG Van de Graaff Generators from $139.95

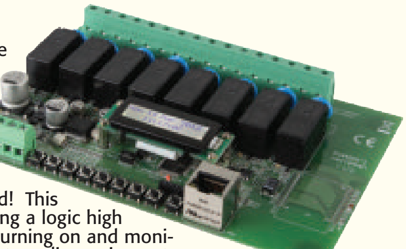
ramsey
www.ramseykits.com

There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit ***www.ramseycatalog.com***

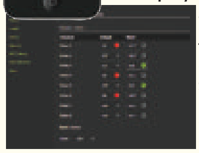
Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

8-Channel Remote Ethernet Controller



Now you can easily control and monitor up to 8 separate circuits via the standard Ethernet network in your home or office. Connection wise it couldn't be simpler. The controller functions as an IP based web server, so it can be controlled by any internet browser that can reach your network! There are no drivers or proprietary software required, just access the controller like any web page from your PC, laptop, or even your smartphone! Security is assured allowing up to 4 separate user credentials. The controller can be set to a specific static IP within your network subnet or can be set to DHCP (auto negotiate). The controller can even be programmed to send you an email to notify and confirm power up and status changes!



To simplify the connection of your equipment to the controller, 8 separate and isolated relay outputs are provided! This gives you internet or network control of up to 8 separate functions. No need to worry about interfacing a logic high or logic low, or burning up the interface! The applications are endless! From something as simple as turning on and monitoring lights at your house with a normal latched closure to advanced control of your electronic gadgets, radio equipment, or even your garage door! Each relay contact is rated at 12A at 30VDC or 16A at 230VAC. Each of the 8 channels has built-in timer and scheduler programs for day, weekend, working days, every day, and every day except Sunday. Relay control functions are programmable for on, off, or pulse (1-99 seconds, 1-99 minutes, or 1-99 hours). In addition to control functions, the web interface also displays and confirms the status of each channel. Each channel can be custom labeled to your specific function name. The controller operates on 12VDC or 12VAC at 500mA or our new AC121 global 12VDC switching power supply below. Factory assembled, tested, and ready to go! Even includes a Cat-5 cable!

VM201 8-Channel Remote Ethernet Controller, Factory Assembled & Tested $169.95

Laser Beam Audio Communicator

Now you can talk to your friends over one of the most secure long-distance transmission types available, a laser beam! The transmitter uses a microphone or external audio to modulate a laser beam on and off at a rate of more than 16kHz so the audio fidelity is much better than that of a standard 3kHz telephone line! And being an optical path, there certainly won't be any 3-letter agencies or anyone else listening!

The receiver includes filtering to remove the 16kHz carrier and leave behind the high quality audio, and then boost its level for use with earphones. Transmitter audio AGC keeps your level perfect! Includes transmitter, receiver and laser pointer. Each runs on a 9V battery (not included).

LBC6K Laser Beam Audio Communicator Kit $59.95

Signal Magnet Antenna

The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.

SM100 Signal Magnet Antenna Kit $89.95

Audio Recorder & Player

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

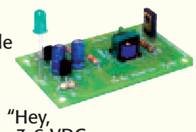
K8094 Audio Recorder/Player Kit $32.95

Tone Encoder/Decoder

Encode and decode with the same kit! This little mini-kit will simultaneously encode and/or decode any audio frequency between 40Hz and 5,000 Hz! Precision 20-turn trim pot adjustment! 5-12VDC.

TD1 Tone Encoder/Decoder Kit $9.95

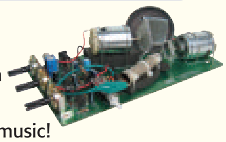
Tickle-Stick Shocker



The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 Tickle Stick Kit $9.95

Laser Light Show



Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 Laser Light Show Kit $49.95

Optically Isolated Module

The hobbyist's headache solver! Converts any AC or DC signal to logic level. The beauty is that the input and output are totally isolated from each other! Output can drive up to 150mA at 40VDC.

OM2 Optically Isolated Module Kit $16.95

Water Sensor Alarm

This little $7 kit can really "bail you out"! Simply mount the alarm where you want to detect water level problems (sump pump)! When the water touches the contacts the alarm goes off! Sensor can even be remotely located. Runs on a standard 9V battery.

MK108 Water Sensor Alarm Kit $6.95

12VDC Regulated

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores for RF and EMI sup-

AC121 12VDC 1A

Classic Nixie Tube Clocks *HOT SELLER!*

Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more!

The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

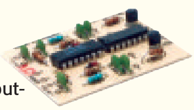
We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment.

Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of!

The clocks are available in our signature hand rubbed Teak & Maple or clear acrylic bases. And for the 6-tube clocks you also have your choice of IN-14 nixie tubes (5=upside down 2!) or highly sought after IN-8-2 nixie tubes.

NIXIE Classic Nixie Tube Clock Kits From $229.95

Touch Switch



The ultimate touch switch! Touch once - it's on, touch again - it's off, or use the momentary outputs that stay on only as long as touched. Two switch circuits on each board. Drives loads up to 100mA. Runs on 6-12VDC.

TS1 Touch Switch Kit $9.95

12VDC Worldwide

It gets even better than our AC121 above! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility!

PS29 12VDC 1.25A Global Power Supply $19.95

Tri-Field Meter Kit

THE GHOST DETECTOR!

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of

TFM3C Tri-Field Meter Kit $74.95

Electronic Watch Dog

A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity! Unlike my Greyhound, it eats 9-12VDC, it's not fussy!

K2655 Electronic Watch Dog Kit $39.95

Electret Condenser Mic

This extremely sensitive 3/8" mic has a built-in FET preamplifier! It's a great replacement mic, or a perfect answer to add a mic to your project. Powered by 3-15VDC, and we even include coupling cap and a current limiting resistor! Extremely popular!

MC1 Mini Electret Condenser Mic Kit $3.95

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 Sniff-It RF Detector Probe Kit $27.95